Project Summary: Investigating Innovation in Free/Libre Open Source Software Development Teams

We propose a social science study in the context of software development to advance our understanding of fundamental processes of learning, process change and product innovation in distributed teams. Our study addresses the general research question: How do members of distributed technology-supported teams of Free/Libre Open Source Software (FLOSS) developers learn to improve their performance by forming shared mental models, individual roles, informal norms and formal rules and how do these structures guide effective behaviours leading to innovations? This question is important because organizational work is increasingly performed by distributed teams of interdependent knowledge workers. These teams have many benefits, but the distance (geographic, organizational and social) between members challenges team members to create the shared understandings and social structures necessary to be effective. But as yet, research and practitioner communities know little about the dynamics of learning and innovation in distributed teams.

To answer our research question, we will conduct a longitudinal in-depth study identifying and comparing the formation, learning and innovation processes of distributed teams of FLOSS developers. The proposed research will be guided by an advisory board of FLOSS developers to ensure relevance and to help promote diffusion of our findings into practice. We will study how these distributed groups develop shared mental models to guide members' behavior, roles to mediate access to resources, and norms and rules to shape action, as well as the processes by which independent, geographically-dispersed individuals are socialized into teams. As a basis for this study, we develop a conceptual framework that uses a structurational perspective to integrate research on team behaviour, organizational learning, communities of practice and shared mental models. We will utilize qualitative data analysis of team interactions, observation and interview data to investigate these processes. We will also use social network analysis to study the socialization process of members and change in roles over time.

Expected intellectual merits

The study will have theoretical, methodological and practical contributions. Developing an integrated theoretical framework to understand learning in a distributed team will be a contribution to the study of distributed teams. Understanding the interplay of structure and action in these teams is important to improve the effectiveness of FLOSS teams, software development teams, and distributed teams in general. The project will contribute to advancing knowledge and understanding of FLOSS development and distributed work more generally by identifying how these teams learn and innovative and how new members are socialized. The study fills a gap in the literature with an in-depth investigation of the practices adopted by FLOSS teams based on a large pool of data and a strong conceptual framework. As well, we will use several different techniques to analyze the practices, providing different perspectives of analysis and a more reliable portrait of what happens in the development teams.

Expected broader impacts

If successful, the project will benefit society by describing learning for FLOSS development, an increasingly important approach to software development. The study will also shed light on learning in distributed work teams in general, which will be valuable for managers who intend to implement this novel, technology-supported organizational form in practice in organizations. Findings from the study might also be used to enhance the way information and communication technologies (ICT) are used to support distance education or for scientific collaboration, which are emerging applications of distributed teams. In order to improve infrastructure for research, we plan to make the tools and raw data available to other researchers. As well, the project involves an international collaboration. Such exchanges expand the perspectives, knowledge and skills of both groups of scientists. Finally, the project will promote teaching, training, and learning by providing graduate and undergraduate students an opportunity to work in teams, integrate their competencies and develop new skills in data collection and analysis.

Project Description: Investigating Innovation in Free/Libre Open Source Software Development Teams

We propose a mixed-mode (qualitative and quantitative) social science field study in the context of software development to advance our understanding of fundamental processes of organizational learning, team development and product innovation in distributed teams. We will address the general research question:

> How do members of distributed technology-supported teams of Free/Libre Open Source Software (FLOSS) developers learn to improve their performance by forming shared mental models, individual roles, informal norms and formal rules and how do these structures guide effective team behaviours leading to innovations?

The proposed research will include a partnership with FLOSS development teams to provide access to data, to ensure relevance and to help promote diffusion of our findings into practice. As evidenced by the attached letters of support from FLOSS developers, members of the FLOSS community are themselves interested learning how to improving their teams' performance.

Introduction to FLOSS teams as technology-supported organizational form that creates innovations

Revolutionary technologies and ideas have created a more closely linked world with almost instantaneous transmission of information. A prominent example of this transformation is the emergence of FLOSS (e.g., Linux or Apache), software innovations [118] created by users [179-181], other volunteers and professionals working in distributed, loosely-coupled teams. FLOSS is a broad term used to embrace software developed and released under an "open source" license allowing inspection, modification and redistribution of the software's source without charge[1]. There are thousands of FLOSS projects, spanning a wide range of applications. Due to their size, success and influence, the Linux operating system and the Apache Web Server (and related projects) are the most well known, but hundreds of others are in widespread use, including projects on Internet infrastructure (e.g., sendmail, bind), user applications (e.g., Mozilla, OpenOffice) and programming languages (e.g., Perl, Python, gcc). Many are popular (indeed, some dominate their market segment) and the code has been found to be generally of good quality [167].

Key to our interest is the fact that most FLOSS software is developed by distributed teams that include users, volunteers and professionals. These teams are close to pure virtual teams in that developers contribute from around the world, meet face-to-face infrequently if at all, and coordinate their activity primarily by means of computer-mediated communications (CMC) [151,187]. The teams have a high isolation index [147] in that most team members work on their own and in most cases for different organizations (or no organization at all). For FLOSS teams, distributed work is not an alternative to face-to-face: it is the only feasible mode of interaction. As a result, these teams depend on processes that span traditional boundaries of place and ownership. FLOSS teams are distributed collaborative structures, not bounded by traditional organizational identity or psychological employment contracts. Thus, a FLOSS team is an example of a *process innovation* [150] that improves the creation of a *technology innovation* (software systems) [113].

The research literature on software development and on distributed work emphasizes the difficulties of distributed software development, but FLOSS development presents an intriguing counter-example. What is perhaps most surprising about the FLOSS process is that it appears to eschew traditional project coordination mechanisms such as formal planning, system-level design, schedules, and defined development processes [9,93]. As well, many (though by no means all) programmers contribute to projects as volunteers, without being paid. Characterized by a globally distributed developer force and a rapid and reliable software development process, ef-

[1] Much (though not all) of this software is also "free software", meaning that derivative works must be made available under the same unrestrictive license terms ("free as in speech", thus "libre"). We use the acronym FLOSS rather than the more common OSS to emphasize this dual meaning.

fective FLOSS development teams somehow profit from the advantages and overcome the challenges of distributed work [5]. The "miracle of FLOSS development" poses a real puzzle and a rich setting for researchers interested in the work practices of distributed teams.

As well, FLOSS development is an important phenomena deserving of study for itself [64]. FLOSS is an increasingly important commercial phenomenon involving all kinds of software development firms, large, small and startup. Millions of users depend on systems such as Linux and the Internet (heavily dependent on FLOSS tools), but as Scacchi [163] notes, "little is known about how people in these communities coordinate software development across different settings, or about what software processes, work practices, and organizational contexts are necessary to their success".

The remainder of this proposal is organized into four sections. In section 1, we present the research setting and discuss the challenges faced by FLOSS teams in learning and innovation. In section 2, we develop a conceptual framework for our study, drawing on theories of shared mental models [24,189] and organizational [97,121] and team learning [61], and using structuration theory [14] as an organizing framework. In section 3, we present the study design, with details of the data collection and analysis plans. We conclude in section 4 by sketching the intellectual merits and expected broader impacts of our study and reviewing the results of prior support.

1. The challenge of distributed software development

Distributed teams are groups of geographically dispersed individuals working together over time towards a common goal. Though distributed work has a long history [e.g., 139], advances in information and communication technologies have been crucial enablers for recent developments of this organizational form [3] and as a result, distributed teams are becoming more popular [126]. Distributed teams seem particularly attractive for software development because the code can be shared via the systems used to support team interactions [137,162]. While distributed teams have many potential benefits, distributed workers face many real challenges. Watson-Manheim, Chudoba, & Crowston [186] argue that distributed work is characterized by numerous discontinuities: a lack of coherence in some aspects of the work setting (e.g., organizational membership, business function, task, language or culture) that hinders members in making sense of the task and of communications from others [177], or that produces unintended information filtering [53] or misunderstandings [8]. These interpretative difficulties in turn make it hard for team members to develop shared mental models of the developing project [52,62]. A lack of common knowledge about the status, authority and competencies of team participants can be an obstacle to the development of team norms [12] and conventions [124].

The presence of discontinuities seems likely to be particularly problematic for software developers [177], hence our interest in distributed software development. Numerous studies of the social aspects of software development teams [51,98,161,177,185] conclude that large system development requires knowledge from many domains, which is thinly spread among different developers [51]. As a result, large projects require a high degree of knowledge integration and the coordinated efforts of multiple developers [19]. More effort is required for interaction when participants are distant and unfamiliar with each others work [141,165]. The additional effort required for distributed work often translates into delays in software release compared to traditional face-to-face teams [94,129]. The problems facing distributed software development teams are reflected in Conway's law, which states that the structure of a product mirrors the structure of the organization that creates it. Accordingly, splitting software development across a distributed team will make it hard to achieve an integrated product [93].

In response to the problems created by discontinuities, studies of distributed teams stress the need for significant time spent learning how to communicate, interact and socialize using computer-supported communications tools [21]. Research has shown the importance of formal and informal coordination mechanisms and information sharing [185] for a project's performance and quality. Communication can help clarify potential uncertainties and ambiguities and socialize members with different cultures and approaches into a cohesive team [77,92,99,102,105].

Successful distributed teams share knowledge and information and create new practices to meet the task and social needs of the members [154].

Research on FLOSS development

The nascent research literature on FLOSS has addressed a variety of questions. Firstly, researchers have examined the implications of FLOSS from economic and policy perspectives. For example, some authors have examined the implications of free software for commercial software companies or the implications of intellectual property laws for FLOSS [e.g., 55,104,116]. Secondly, various explanations have been proposed for the decision by individuals to contribute to projects without pay [e.g., 15,85,86,95,125]. These authors have mentioned factors such as increasing the usefulness of the software [86], personal interest [86], ideological commitment, development of skills [119] with potential career impact [86] or enhancement of reputation [125].

Thirdly, a growing stream of research examines factors for the success of FLOSS in general terms. The popularity of FLOSS has been attributed to the speed of development and the reliability, portability, and scalability of the resulting software as well as the low cost [45,83,115,149,150,174,175]. In turn, the speed of development and quality of software have been attributed to two factors: that developers are also users of the software and the availability of source code. Firstly, FLOSS projects often originate from a personal need [132,178], which attracts the attention of other users and inspire them to contribute to the project. Since developers are also users of the software, they understand the system requirements in a deep way, eliminating the ambiguity that often characterizes the traditional software development process: programmers know their own needs [106]. Secondly, in FLOSS projects, the source code is open to modification, enabling users to become co-developers by developing fixes or enhancements. As a result, FLOSS bugs can be fixed and features evolved quickly. Asklund & Bendix [9] note the resulting importance of well-written and easy-to-read code.

Finally, a few authors have investigated the detailed processes of FLOSS development [e.g., 72,100,151,169], which is the focus of this proposal. Raymond's [151] bazaar metaphor is perhaps the most well-known model of the FLOSS process, though it is not without critics [16]. As with merchants in a bazaar, FLOSS developers are said to autonomously decide how and when to contribute to project development. By contrast, traditional software development is likened to the building of a cathedral, progressing slowly under the control of a master architect. Recent empirical work has begun to better illuminate the structure and function of FLOSS development teams. Most of these studies have been case studies focused on a few large projects [e.g., 66,108,112,114,130,133]; there have been fewer comparisons across projects [e.g., 75,107,170].

The PI on this proposal, Kevin Crowston, has been active in FLOSS research, supported by NSF grant IIS 04–14468 ($327,026, 15 August 2004 to 14 August 2006), for *Effective work practices for Open Source Software development*, which continued SGER IIS 03–41475, ($12,052, 1 September 2003 to 31 August 2004). The initial results of this funding include an analysis of FLOSS teams as virtual organizations [45], theoretical models of FLOSS team effectiveness [32,34] and leadership [35] and a study of possible success measures for FLOSS [31,33]. Empirically, we have analyzed the problems in using SourceForge data [96], carried out social network analyses of centralization and hierarchy of project teams [36,38] and described the role of face-to-face meetings in FLOSS teams [39]. The earlier grant was aimed at identifying work practices that characterize effective FLOSS teams. In the research proposed here, we seek to extend this prior work by examining how teams learn to be effective and to innovate.

We have chose this new focus because studies of FLOSS teams (including our own) and of distributed teams more generally point to the need to understand the novel organizational form of technology-supported self-organizing distributed groups and the processes through which these organizations learn to improve their performance. In their study, Robey et al. [154] suggest that to be successful, distributed teams must share knowledge and information and create new practices to meet the task and social needs of the members of the team. More generally, an organization's capability to learn has been recognized as a core competency necessary for survival and competition in a knowledge-based economy [70,121]. The better an organization is at learning, the better it can be at adapting to the environment, correcting for error, and innovating [7]. Accordingly, to minimize the negative effects of being distributed, FLOSS teams have to learn to communicate, coordinate and create a cohesive whole. However, research and practitioner communities know little about the processes of knowledge sharing, learning and socialization suitable for distributed teams [144,154]. Thus it is important for us to first understand these teams and their learning processes. As Maier et al. say, "Knowledge about the process, or the know how, of learning facilitates corrections that simulate or accelerate learning" [121].

2. Conceptual development

In this section we develop the theoretical framework for our study, building on and adding to existing literature drawn from multiple disciplines. For this project, we have chosen to analyze developers as comprising a work team. Much of the literature on FLOSS has conceptualized developers as forming communities, which is a useful perspective for understanding why developers choose to join or remain in a project. However, for the purpose of this study, we view the projects as entities that have a goal of developing a product, whose members are interdependent in terms of tasks and roles, and who have a user base to satisfy, in addition to having to attract and maintain members. These teams have a social identity, in which core members of the projects know and acknowledge each other's contributions. These aspects of FLOSS projects suggest analyzing them as work teams. Guzzo and Dickson [81] defined a work team as "individuals who see themselves and who are seen by others as a social entity, who are interdependent because of the tasks they perform as members of a group, who are embedded in one or more larger social system (e.g., community, or organization), and who perform tasks that affect others (such as customers or coworkers)". Although there are substantial similarities, as discussed below, we argue that FLOSS teams are more than communities of practice because members have a shared output; in communities of practice (e.g., the copier repairmen studied by Orr [146]), members share common practices, but are individually responsible for their own tasks. We have further chosen to focus on the teams' practices because of our focus on how teams learn and innovate.

A structurational perspective on team dynamics

To conceptualize the dynamics of these teams and the process of learning within them, we adopt a structurational perspective. Structuration theory [74] is a broad sociological theory that seeks to unite action and structure and to explain the dynamic of their evolution. Numerous authors have used a structurational perspective to support empirical analyses of group changes [13,54,138,142,183], though a discussion of the merits of each use is beyond the scope of this proposal. Here, we build on the view of structuration presented by Orlikowski [142] and Barley and Tolbert [14]. We chose this framework because it provides a dynamic view of the relations between team and organizational structures and the actions of those that live within, and help to create and sustain, these structures. The theory is premised on the duality of structures, that is, systems of signification, domination and legitimation that influence individual action. In this view, structure is recursive: the structural properties of a social system are both the means and the ends of the practices that constitute the social system. As Sarason [158] explains, in structuration theory:

> "The central idea is that human actors or agents are both enabled and constrained by structures, yet these structures are the result of previous actions by agents. Structural properties of a social system consist of the rules and resources that human agents use in their everyday interaction. These rules and resources mediate human action, while at the same time they are reaffirmed through being used by human actors or agents." (p. 48).

Simply put, by doing things, we create the way to do things.

By relating structure and function across time, structuration theory provides a framework for understanding the dynamics of a team as it learns [79]. Barley and Tolbert [14] note that structuration is "a continuous process whose operations can be observed only through time" (p. 100). Figure 1, adapted from [14] shows the relation between institution (which the authors use syn-

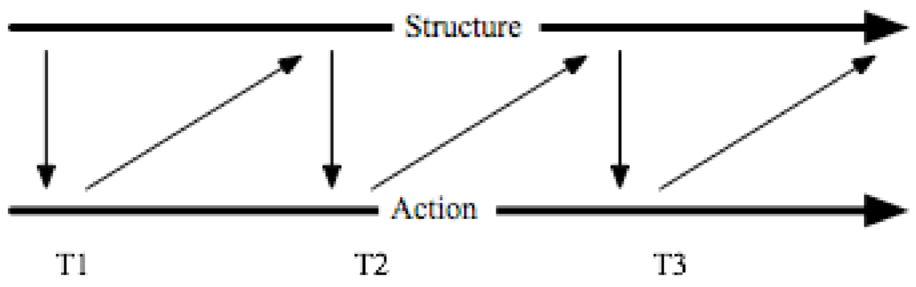


Figure 1. A sequential model of the relation between structure and action [from 14].

onymously with structure) and action, and how both evolve over time. In this figure, the two bold horizontal lines represent "the temporal extensions of Giddens' two realms of social structure: institutions and action," while the "vertical arrows represent institutional constraints on action" and the diagonal arrows, "maintenance or modification of the institution through action" (p.100). As Cassell [29] says, "to study the structuration of a social system is to study the ways in which that system, via the application of generative rules and resources, in the context of unintended outcomes, is produced and reproduced through interaction" (p. 119). Thus, our analysis will describe current team practices (the lower arrow) and current team structures (the upper arrow) and how these interact (the vertical and diagonal arrows) and change over time. In order to explain how the teams are evolving, we present the changes as states or stages (e.g., T1, T2 and T3 in the figure) and highlight the "dislocation of routines" and other temporal disruptions that lead to these different states [79].

Corporate participation and the process of structuration

The structuration perspective also makes clear the importance of any initial structures that individual team members bring from prior experiences (i.e., from an unseen T0 in Figure 1). Barley and Tolbert [14] note that "actors are more likely to replicate scripted behaviours" than to develop new ones. Orlikowski and Yates [145] argue similarly, suggesting that in a new situation individuals will typically draw on their existing repertoires of actions, reproducing those they have experienced as members of other communities. These prior experiences will provide an initial set of structures that guide behaviours, which will be particular important during the formative stages of the team. Because of the importance of these initial structures, we are particularly interested in the effects of corporate participation on FLOSS teams. An interesting recent trend in FLOSS development is the increasing participation by commercial companies in FLOSS development (a form of private/non-profit partnership) [e.g., in the Gnome project, 73]. We hypothesize that teams with strong corporate participation will adopt structures from the surrounding corporate milieu, thus influencing their evolution. The importance of corporate participation is reinforced by other research. For example, Hackman's [82] model of group performance suggests organizational context as an important factor affecting team processes. Finholt and Sproull [67] found that teams who do not work within a specific organizational context have a greater need for team learning. These results have been also been supported by our initial interviews with FLOSS developers, who see corporate participation having an important contribution to team processes and activities.

Conceptualizing structuration in FLOSS teams

To apply structuration as a perspective to conceptualize learning and innovation in distributed FLOSS teams, we first must clarify the types of rules and resources that comprise the structure. For this work, we specifically consider three kinds of rules and resources that are "encoded in actors' stocks of practical knowledge" and "instantiated in recurrent social practice" [143]: interpretive schemes, resources, and norms [14,168]. In the remainder of this section, we elaborate each of these three aspects of structure as they apply to FLOSS development in particular. We note that all of these issues apply as well in physically proximal teams but are more difficult to manage in the dispersed/distributed teams that are our focus.

Interpretive schemes and structures of signification. Individual actors' interpretive schemes create structures of *signification* and thus influence (and are created by) individual actions. To describe how these schemes influence action and vice versa, we draw on the literature on the role of shared mental models in team action. Shared mental models, as defined by Cannon-Bowers et al. [23], "are knowledge structures held by members of a team that enable them to form accurate explanations and expectations for the task, and in turn, to coordinate their actions and adapt their behavior to demands of the task and other team members" (p. 228). Shared mental models are thus related to transactive memory [91], which describes how individuals remember where to find information. That theory was originally developed to explain the behaviours of intimate couples, but recently extended to groups [131] and distributed teams [80,122]. However, research indicates that transactive memory converges to shared mental models as "individuals develop a shared conceptualization of 'who knows what.'" [18]. Yoo & Kanawattanachai [192] similarly argues that transactive memory can develop to collective mind [189]. In our work, we therefore build on the broader concept.

Research suggests that shared mental models help improve performance in face-to-face [153] and distributed teams [171]. Shared mental models can enable teams to coordinate their activities without the need for explicit communications [40,63]. Without shared mental models, individuals from different teams or backgrounds may interpret tasks differently based on their backgrounds, making collaboration and communication difficult [56]. The tendency for individuals to interpret tasks according to their own perspectives and predefined routines is exacerbated when working in a distributed environment, with its more varied individual settings. Research on software development in particular has identified the importance of shared understanding in the area of software development [117,157]. Curtis et al. [52], note that, "a fundamental problem in building large systems is the development of a common understanding of the requirements and design across the project team." They go on to say that, "the transcripts of team meetings reveal the large amounts of time designers spend trying to develop a shared model of the design". The problem of developing shared mental models is likely to particularly affect FLOSS development, since FLOSS team members are distributed, have diverse backgrounds, and join in different phases of the software development process [60,71].

In emphasizing the duality of structure, the structurational perspective draws our attention to how shared mental models are products of, as well as guides to, action. Walton and Hackman [184] identify an interpretive function of teams, which is to help members create a consistent social reality by developing shared mental models. To identify specific actions that can help to build shared mental models, we turn to Brown and Duguid [20], who identify the importance of socialization, conversation and recapitulation in communities of practice. Firstly, new members joining a team need to be socialized into the team to understand how they fit into the process being performed through a process of socialization, e.g., by following a "joining script" [182]. Members need to be encouraged and educated to interact with one another to develop a strong sense of "how we do things around here" [91]. Barley and Tolbert [14] similarly note that socialization frequently "involves an individual internalizing rules and interpretations of behaviour appropriate for particular settings" (p. 100). Secondly, conversation is critical in developing shared mental models. It is difficult to build shared mental models if people do not talk to one another and use common language [117]. Meetings, social events, hallway conversations and electronic mail or conferencing are all ways in which team members can get in touch with what others are doing and thinking (though many of these modes are not available to distributed teams). Finally, Brown and Duguid [20] stress the importance of recapitulation. To keep shared mental models strong and viable, important events must be "replayed", reanalyzed, and shared with newcomers. The history that defines who we are and how we do things around here must be continually reinforced, reinterpreted, and updated.

Resources and structures of domination. The control of resources is the basis for power and thus for structures of *domination*. For software development, material resources would seem to be less relevant, since the work is intellectual rather than physical and development tools are readily available, thanks to openly available FLOSS development systems such as SourceForge

[10, http://sourceforge.net/] and Savannah (http://savannah.gnu.org/). Furthermore, most FLOSS teams have a stated ethos of open contribution. However, team members face important differences in access to expertise and control over system source code in particular. To understand the functions of these resources, we plan to examine different roles in the software development process and how they affect individual contributions, and how these roles are established and maintained.

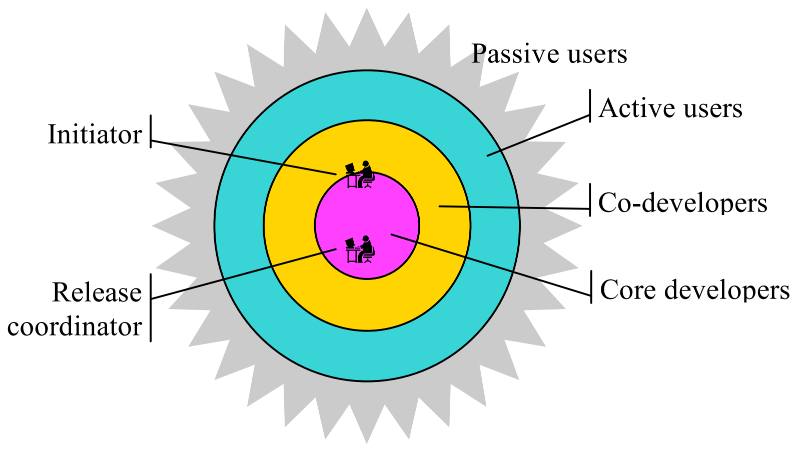


Figure 2. Hypothesized FLOSS development team structure.

Several authors have described FLOSS teams as having a hierarchical [164] or onion-like structure [30,68,133,155], as shown in Figure 2. At the centre are the core developers, who are usually distinguished by having write privileges or other formal authority over the source code [75,76]. Core developers contribute most of the code and oversee the design and evolution of the project. The core is usually small (e.g., 9 [101], 11 [103] or 15 [129] developers) and exhibits a high level of interaction (most developers know and acknowledge each other), which would be difficult to maintain if the core group were large. Surrounding the core are perhaps ten times as many co-developers. These individuals contribute sporadically by reviewing or modifying code or by contributing bug fixes. The co-developer group can be much larger than the core, because the required level of interaction is much lower. The apparent reliance of FLOSS teams on this structure provides an interesting contrast to conventional teams: in a study of 182 work teams, Cummings and Cross [50] found that core-periphery and hierarchical team structures were negatively associated with performance. On the other hand, Halloran & Scherlis [84] suggest that FLOSS processes allow co-developers to move in and out of the project without hampering its function. Surrounding the developers are the active users: a subset of users who use the latest releases and contribute bug reports or feature requests (but not code). Still further out are passive users, who use the project's outputs but are not otherwise part of the project. Giuri et al. [75] similarly distinguished empirically between project managers, developers and manager-developers and others, with manager-developers most involved and others least.

There is some evidence that clear definition of these roles is important for project effectiveness. Sutanto, Kankanhalli & Tan [171] found that role ambiguity in distributed teams led to duplicate work (though this is often not considered a problem in FLOSS teams). Sagers [156] argues that restricting access to the core improves coordination and success of project. Halloran & Scherlis [84] similarly argue for a "walled server" to manage the in-flow of information. It is also important that various roles be filled. Active users in particular play an important role in FLOSS development [148]. Research suggests that more than 50 percent of the time and cost of non-FLOSS software projects is consumed by mundane work such as testing [166]. The FLOSS process enables hundreds of people to work on these parts of the process [114], what Rossi [155] describes as "parallel development… enabled by the modularization of the source code". Giuri et al. [75] found that the share of external contributors had a positive impact on project success. Koch & Schneider [103] state bluntly, "the attraction of participants is therefore identified as one of the most important aspects of open source development projects."

However, how roles are defined and maintained within a project is still an open question. Prior case studies have described how individuals move from role to role as their involvement with a project changes. For example, a common pattern is for active users to join the core development team in recognition of their contributions and ability. In some teams, this selection is an informal process managed by the project initiator, whiles others such as the Apache Project have formal voting processes for vetting new members. However, core developers must have a deep understanding of the software and the development processes, which poses a significant barrier to entry, particular in a distributed team [65,89]. This barrier is troubling because of the reliance of FLOSS projects on volunteer submission and "fresh blood" [49]. On the other hand, we are still learning how the privileges and responsibilities of these different roles are defined. Again, some projects seem to have formal role definitions, while in others, roles seem to be more emergent.

Rules and norms and structures of legitimation. Finally, actors' social norms and team rules embody structures of *legitimation*. The regulative function of teams, as presented by Walton and Hackman [184], describes one aspect of team functions as the creation of implicit norms and explicit rules [172]. The importance of such rules have been documented in conventional software and FLOSS development teams [e.g., 159,170]. Rossi [155] notes that rules allow developers to form stable expectations of others actions, thus promoting coordination. For example, Jørgensen [101] describes a set of implicit and explicit rules for software development in the FreeBSD project (e.g., "Don't break the build"), while Raymond [152] notes implicit rules regarding project forking at the community level. Gallivan [69] analyzes descriptions of the FLOSS process and suggests that teams rely on a variety of social control mechanisms rather than on trust.

In our discussion above of shared mental models, we noted the importance of socialization, which helps to spread norms as well as beliefs. However, consideration of structures of legitimation raises the question of the origin of rules and norms. As the team attempts to achieve its task, team interactions lead to the development of implicit and explicit rules for social or interpersonal interaction to guide team member behavior in achieving its goals and functions. These changes are the result of integrating the knowledge of experts into the team's structure reflecting behavioral changes within a team over time, what March et al. [123] and Hayes and Allinson [88] refer to as learning on the group level. Grant [78] similarly suggests that a firm (or team) creates coordination mechanisms, in the form of procedures and norms, to economize on communication, knowledge transfer and learning, thus reserving team decision making and problem solving for complex and unusual tasks. However, the practices by which these rules can be developed in distributed settings is an open issue.

3. Research Design

In this section, we will discuss the design of the proposed study, addressing the basic research strategy, concepts to be examined, sample populations and proposed data collection and analysis techniques. In this section, we first discuss the goals and general design of the study. We then present the details of how data will be elicited and analyzed.

Longitudinal multiple case study of four FLOSS teams

To study the processes through which FLOSS teams learn to improve their performance and carry out effective innovation, we will carry out a longitudinal in-depth multiple case study design, as suggested by Barley and Tolbert [14]. Many authors have noted that only a small number of FLOSS projects are truly active and thus likely to develop significant innovations [75], so we chose a research strategy that will provide rich detail about a small number of projects. Furthermore, working closely and in-depth with four development team partners will allow us to get deep inside the innovation processes that develop in these organizations.

In this section we present the overall research design, shown in Figure 3, followed by details of the data elicitation and analysis. The overall research design follows the plan laid out by Barley and Tolbert [14], who suggest four steps in a study to investigate the dynamics of structure:

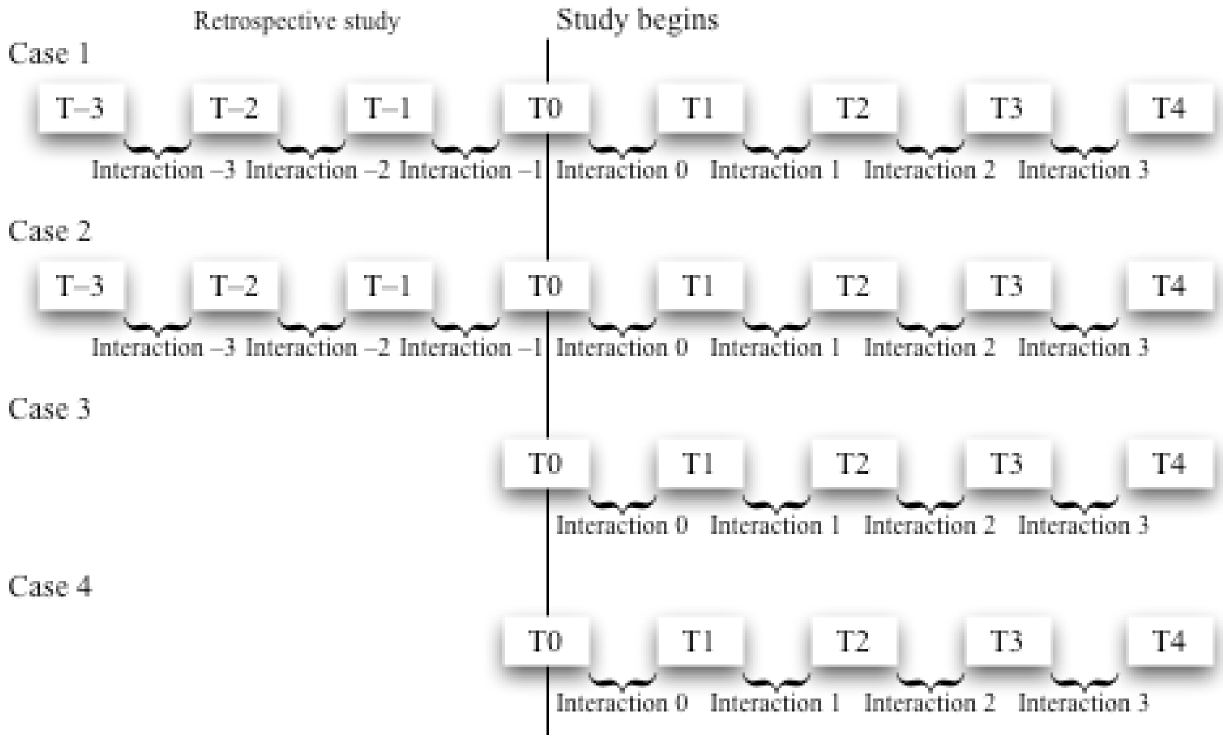
> "(1) defining an institution (structure) at risk of change over the term of the study and selecting sites in light of this definition; (2) charting flows of action at the sites and extracting scripts characteristic of particular periods of time; (3) examining scripts for evidence of change in behavioral and interaction patterns; and (4) linking findings from observational data to other sources of data on changes in the institution of interest" (pg. 103).

We will next discuss how we implement each of these steps, while deferring discussion of the details of data collection and analysis to subsequent sections.

Step one: Selecting sites. We will start by identifying promising projects for investigating the dynamics of structure and action. We plan to study four FLOSS project teams in depth to allow for comparison on two dimensions. For some of the cases (cases 1 and 2 in Figure 3), we will combine the longitudinal study with retrospective data analysis. In selecting teams to study, we will consider theoretical and pragmatic aspects.

- Firstly, we will compare projects that vary in their level of corporate participation, for the reasons discussed above in the conceptual development section.
- Secondly, we will compare two newly-formed and two well-established project teams. We will study the development of the teams longitudinally and the two established teams retrospectively as well. Picking newly-formed teams will allow us to study the initial stages of team formation and in particular the negotiation among previously experienced structures brought in by team members. However, relying entirely on new teams seems risky. Firstly, Barley and Tolbert [14] note the difficulties of identifying settings that are likely to experience interesting changes. Secondly, we want to ensure that we study some teams that have developed effective work practices. Studying some established teams allows us to choose some projects that are known to be effective. Studying established projects also enables study of the processes of socialization of new members into an on-going project.
- Thirdly, in order to ensure that we are studying teams large enough to have coordination problems (as opposed to single person development efforts [107]), we will choose only projects with more than seven core developers [87].
- Finally, in selecting projects, we will also have to take into consideration some pragmatic considerations. We will select FLOSS teams where we have access to the data we need (e.g., message logs) and where we can obtain the participation of developers for interviews. We are fortunate to have already obtained agreements to cooperate from leading FLOSS developers, as shown by the attached letters of support included in the supporting documents.

Figure 3. Overall research design.



Step two: Charting flows of actions. In this step we extract the interactions of team members within a particular time period to investigate how the teams learn and develop over time. We plan to interview developers for each case at least every six months (T1, T2, T3 and T4 in Figure 3). Six months was chosen since it provides a small enough gap to be able to trace the process of change, relying on developers' memories of events, while still being feasible for data collection and not too onerous for participants. We will also extract team interactions from email logs, ethnographic field notes, and observations of developer activities between the six month measurement points to analyze the dynamics that lead to the observed changes. For two of the cases, we will carry out a similar analysis on retrospective data (potentially over the entire recorded history of the project). The details of data elicitation and analysis are discussed in the following sections.

Step three: Identifying patterns of changes. Once we extract the segments of interactions discussed in step two, we will analyze the interaction to uncover patterns of behavior through which members change shared mental models, roles, and norms and rules. We will investigate the processes by which teams develop shared mental models by studying how members contribute to and coordinate tasks, paying special attention to evidence of socialization, conversation and recapitulation. We will study how roles are assigned and evolve over time by studying member contribution and by looking for evidence of role definition and role changes. Lastly, we will study the dynamics by which rules and norms evolve by also looking for task contribution and coordination, paying special attention to evidence of rules creation and modification.

Step four: Linking changes in structures to other changes. In Step 4, Barley and Tolbert [14] suggest linking changes in the structures to other changes of interest in the sites being studied. Since the primary focus of our study is the learning and innovation of the teams, this step will not be the major focus of our efforts. Nevertheless, we will triangulate evidence gathered from multiples sources of evidence about the teams. For example, comparisons across the teams will provide evidence to help us understand the role of corporate participation in the teams.

Data collection

To explore the concepts identified in the conceptual development section of this proposal, we will collect evidence from a wide range of data: project demographics, developer demographics, interaction logs, project plans and procedures, developer interviews, and project observation. In the remainder of this section, we will briefly review each source. Table 1 shows the mapping from each construct to data sources.

Developer demographics. We will collect basic descriptive data about developers, such as area of expertise, formal role, years with the project, other projects the developer participates in etc. Often these data are self-reported by the developers on project pages; in other cases, they can be elicited from the developers during interviews. We will track changes in the formal roles of members using this source. By examining PGP key signatures, we can identify meetings between developers [140], which will suggest past opportunities for socialization.

Project plans and procedures. Many projects have stated release plans and proposed changes. Such data are often available on the project's documentation web page or in a "status" file used to keep track of the agenda and working plans [49]. For example, Scacchi [163] examined requirements documentation for FLOSS projects. We will also examine any explicitly stated norms, procedures or rules for taking part in a project, such as the process to submit and handle bugs, patches or feature request. Such procedures are often reported on the project's web page (e.g., http://dev.apache.org/guidelines.html). We will track changes in the various versions of any specific set of rules and procedures.

Interaction logs. The most voluminous source of data will be collected from archives of CMC tools used to support the team's interactions for FLOSS development work [94,114]. These data are useful because they are unobtrusive measures of the team's behaviours [188]. Mailing list archives will be a primary source of interaction data that illuminates the 'scripts' for the analysis of dynamics [14], as email is a primary tool used to support team communication, learning and socialization [113]. Such archives contain a huge amount of information (e.g., the

LINUX kernel list receives 5-7000 messages per month, the Apache HTTPD list receives an average of 40 messages a day). While in most cases these archives are public, we plan to consult with the Syracuse University Human Subjects Institutional Review Board to determine what kind of consent should be sought before proceeding with analysis. From mailing lists, we will extract the date, sender and any individual recipient' names, the sender of the original message, in the case of a response, and text of each message. In a similar analysis of student messages, Dutoit & Bruegge [58] found relations between level, pattern and content of messages and team performance. In addition to email, we will examine features request archives and logs from other interaction tools, such as chat sessions.

Observation. We have found from our current study that developers interact extensively at conferences [39]. Indeed, Nardi and Whittaker [136] note the importance of face-to-face interactions for sustaining social relations in distributed teams. The FreeBSD developer Poul-Henning Kamp has also stated that phone calls can be occasionally used to solve complex problems [60]. These interactions are a small fraction of the total, but they may still be crucial to understanding the team's practices. We plan to use attendance at developer conferences (e.g., the *O'Reilly Open Source Convention* or *ApacheCon*) as an opportunity to observe and document the role of face-to-face interaction for FLOSS teams.

Developer interviews. While the data sources listed above will provide an extensive pool of data, they are mostly indirect. Interviews will provide rich, first-hand data about developers' perceptions and interpretations. We plan to conduct interviews with key informants in the projects. Interviews will be conducted by e-mail and face-to-face at FLOSS conferences. We will explore the developer's initial experiences of participation in FLOSS, the social structure and norms of the team, processes of knowledge exchange and socialization (especially the role of observation or lurking, which leaves no traces in the interaction logs), and knowledge of other members' par-

Table 1. Constructs, sources of data, and analysis.

Structure	**Constructs**	**Data sources (see section 3)**
Signification	Shared mental models	Content analysis of interactions, interviews and observation
	Task coordination and contribution	Process mapping, social network analysis
	Socialization Conversation Recapitulation	Content analysis of interactions, interviews and observation
Domination	Roles with differential access to resources	Process mapping, social network analysis Content analysis of interactions, interviews and observation
	Task coordination and contribution	(See above)
	Role definition Role changes	Process mapping, social network analysis
Legitimation	Norms Formal rules and procedures	Content analysis of interactions, interviews and observation Project plans and procedures
	Task coordination and contribution	(See above)
	Rule creation and change	Content analysis of interactions, interviews and observation

ticipation [134,190]. As well, interviews will be used to verify that the archives of interaction data give a fair and reasonably complete record of day-to-day interactions.

Data analysis

While voluminous, the data described above are mostly at a low level of abstraction. The collected data will be analyzed using a variety of techniques in order to raise the level of conceptualization to fit the theoretical perspectives and constructs described in Section 2 and to document the flows of action (Step 2) and patterns of change (Step 3) that address our research question. Table 2 shows the mapping from data sources to data analysis techniques. We expect the analysis to paint a picture of each project in terms of the contributions towards effective software development of structures of shared mental models, roles and formal and informal rules, and more importantly, the practices in each project that build and evolve these structures as team members learn to work together and to innovate more effectively.

Content analysis. The project will rely heavily on content analysis of the text from interaction archives and interviews to develop insights on the extent and development of shared mental models, rules and norms as well as socialization (e.g., the way projects are created, introduction of new members, members leaving and community building). Analysis of the developers' discourse in various fora will allow us to closely observe how they integrate research and development tasks to carry out effective innovation.

Qualitative data will be content analyzed following the process suggested by Miles and Huberman [127], iterating between data collection, data reduction (coding), data display, and drawing and verifying conclusions. The researchers will work from an initial content analytic framework based on frameworks previously used to investigate shared mental models [e.g., 59]. In addition we will incorporate work on Asynchronous Learning Networks investigating social, cognitive and structuring processes of virtual teams [90]. We will start the data analysis using the initial content analytic scheme and modify the scheme as new categories and indicators emerge in the data [127]. Further categories will be added and other data will be collected as preliminary findings in the analysis suggest. We will use the thematic unit of analysis while conducting the content analysis to capture the various elements of the variables under investigation as appropriate. To increase the validity and reliability of the coding scheme we will conduct intercoder reliability tests and modify the content analytic scheme until we reach an 85% agreement level [11]. While we can accomplish the project's goals using manual coding, we plan to consult with researchers at Syracuse's Centre for Natural Language Processing about ways to use natural language processing technology to automate some analyses, which would allow us to study more projects. Turner et al. [173] similarly used some simple NLP approaches to analyze bug reports.

Social network analysis (SNA). SNA will be used to analyze patterns of interactions (e.g., who responds to whose email) in order to reveal the structure of the social network of projects and their impact on team outcomes. Madey, Freeh & Tynan [120] applied this technique to connections between projects, but not within projects. Ducheneaut [57] examines interaction patterns, but focused on visualization of the networks. Our work using this approach with

Table 2. Data sources and planned analysis approaches.

Data source	**Analysis approach**
Developer demographics	Statistical
Developer interaction logs	Social network analysis
	Content analysis, process mapping
Project plans and procedures	Content analysis
Developer interviews	Content analysis, process mapping, cognitive mapping
Observation of developer interactions	Content analysis, process mapping, cognitive mapping

interactions in bug fixing logs has revealed that projects display a surprising range of centralizations [37] and most were quite hierarchical [38], similar to the results of Ahuja & Carley [2]. However, these analyses have just scratched the surface. We are particularly interested in using social network information to identify various structural roles in the team (e.g., via blockmodels of interactions) and how individuals fill these roles over time. This analysis of structural roles should provide a useful counterpoint to descriptions of formal roles. As well this analysis will track the socialization of members into the core of the team, and the development and changes in leadership over time. We will assess an individual's centrality and the project's hierarchy, which seems to mediate the effect of role and status on individual performance within virtual teams [3], the way contributions are distributed among developers and the roles assumed by core developers. The results of such analyses will support identification of the social relations patterns and the way such patterns develop and affect team learning and socialization. As such, social network analysis provides a clear lens through which we can observe the impacts of asynchronous communication technology on this new and emergent organizational form.

Process maps. The open source software development *processes* will be mapped based on an inductive coding of the steps involved [191]. For example, to map the bug fixing process, we will examine how various bugs were fixed as recorded in the bug logs, email messages and the code. Van de Ven and Poole [176] describe in detail the methods they used to develop and test a process theory of how innovations develop over time. In the FLOSS setting, Yamauchi et al. [191] coded messages to understand the development processes of two projects, while Bonneaud, Ripoche & Sansonnet [17] analyzed bug report messages for Mozilla to understand the bug fixing practices. Process traces can be clustered using optimal matching procedures [1] to develop clusters of processes. These process descriptions can be enriched with descriptions of the process from developers' reports of critical incidents and of the process in general [42].

In our analyses, we will identify which individuals perform which activities to identify different process roles, thus providing a counterpoint to the SNA roles described above. We will also identify the coordination modes and task assignment practices involved in software maintenance (i.e., the number of features request assigned, types of requests, number and types of spontaneous contributions), the adoption of other formal coordination modes (from the analysis of the written policies regarding contributions to projects), as well as the degree of interdependency among the tasks (based on an analysis of communication patterns among different roles and different contributors). Another question we intend to answer is the extent to which the use of various distributed software development tools (e.g., CVS, bug tracking databases) provides a source of structure for the process [9].

Cognitive maps. Cognitive maps will be developed from interview data to represent and compare the mental models of the developers about the project and project team so as to gauge the degree of common knowledge and the development of shared mental models [26,27,111,135]. We are particularly interested in how these maps evolve over time. Metrics (e.g., number of heads, tails, domain and centrality) provided by existing software packages (e.g., Decision Explorer or CMAP2) and ad hoc developed metrics will be used to analyze and compare the different maps. In particular, the comparisons among different team members' maps will provide insights about the processes through which developers collaboratively learn to improve their organizational performance. We will also derive collective maps for each project. Collective maps usually represent perspectives that are common to all the members of a team. Shared perspectives derive from the comprehension of mutual positions and roles, which are fundamental to create synergies within the team. The PI has some experience studying mental models [40] but for this analysis in particular will work with a collaborator, Professor Barbara Scozzi, as discussed below.

Work plan

Based on preliminary assessment of the effort required, we are requesting funding for two graduate students. The graduate students will devote 50% effort during the academic year and 100% effort during the summers, for a total of 2200 hours/year (4400 hours in two years). The graduate students will support the principal investigators in sample section, definition of constructs and variables, and will have primary responsibility for data collection and analysis for two of the cases each, under the oversight of the PI. The co-PI, Robert Heckman, will work one-third-time on the project during the summers, 1 month per year. Summers will be devoted to sample selection, interviews and publication of results. The PI, Kevin Crowston, will work 1/6 time during the summer on project management and research design. Both PIs will devote 10% of effort during the academic year to project management and oversight (1/2 day / week, supported by Syracuse University).

These activities, in particular those related to the analysis of shared mental models within the FLOSS development teams, will be carried out with the assistance of an international collaborator, Dr. Barbara Scozzi of the Department of Mechanical and Business Engineering, Polytechnic of Bari, Italy (please see the supporting documents section for a letter of support and vitae; no funding is being requested from NSF to support Dr. Scozzi). Dr. Scozzi has collaborated with the PI on a study of FLOSS project success factors [45] and her competencies in cognitive mapping [4,25] will be particularly valuable for this project.

4. Conclusion

In this proposal, we develop a conceptual framework and a research plan to investigate organizational learning that support innovation within distributed FLOSS development teams. To answer our research question, we will conduct a longitudinal in-depth study identifying and comparing the formation and evolution of distributed teams of FLOSS developers. We will study how these distributed groups develop shared mental models to guide members' behavior, roles to control access to resources, and norms and rules to shape action and the dynamics by which independent, geographically-dispersed individuals are socialized into the group.

Expected intellectual merits

The project will contribute to advancing knowledge and understanding of distributed teams by identifying the dynamics of learning and innovation in distributed FLOSS teams. The study has two main strengths. Firstly, we fill a gap in the literature with an in-depth investigation of the processes of developing shared mental models, roles and norms and rules in FLOSS teams and of socializing new members to these structures, based on a large pool of data and a strong conceptual framework. Secondly, we use several different techniques to analyze the learning process, providing different perspectives of analysis and a more reliable portrait of what happens in the development teams. Moreover, some of data analysis techniques, such as cognitive maps and social network analysis, have not yet been used with FLOSS teams.

We expect this study to have theoretical, methodological and practical contributions. Understanding the dynamics of learning in a team of independent knowledge workers working in a distributed environment is important to improve the effectiveness of distributed teams and of the traditional and non-traditional organizations within which they exist. As Maier et al. suggest; "Knowledge about the process, or the know how, of learning facilitates corrections that simulate or accelerate learning" [121]. Developing a theoretical framework consolidating a number of theories to understand the dynamics within a distributed team is a contribution to the study of distributed teams and learning within organization literature [154]. Employing qualitative techniques to understand the process of learning will also be a contribution to the organizational learning methodology [128].

Expected broader impacts

The project has numerous broader impacts. The project will benefit society by identifying the dynamics of learning and socialization in FLOSS development, an increasingly important approach to software development and innovation. The study will also shed light on dynamics of learning and socialization for distributed work teams in general, which will be valuable for managers who intend to implement such an organizational form. Understanding the dynamics of learning and socialization can serve as guidelines (in team governance, task coordination, communication practices, mentoring, etc.) to improve performance and foster innovation. Understanding these questions is important because today's society entails an increased use of distributed teams for a wide range of knowledge work. Distributed work teams potentially provide several benefits but the separation between members of distributed teams creates difficulties in coordination, collaboration and learning, which may ultimately result in a failure of the team to be effective [22,28,99,105]. For the potential of distributed teams to be fully realized, research is needed on the dynamics of learning and socialization. As well, findings from the study can be used to enhance the way CMC technologies are used in education or for scientific collaboration. For example, the results could be used to improve the design and facilitation of e-learning courses and distance classes. Finally, understanding FLOSS development teams may be important as they are potentially training grounds for future software developers. As Arent and Nørbjerg [6] note, in these teams, "developers collectively acquire and develop new skills and experiences".

To ensure that our study has a significant impact, we plan to broadly disseminate results through journal publications, conferences, workshops and on our Web pages. We also plan to disseminate results directly to practitioners through interactions with our advisory board and with developers, e.g., at FLOSS conferences. Our results could also potentially be incorporated into the curricula of the professional masters degrees of the Syracuse University School of Information Studies, as well as improving the pedagogy, as these programs are offered on-line and thus involve distributed teams. Findings about the dynamics of the learning process in FLOSS development teams can also benefit the design of technology and engineering curricula. These fields use similar processes for learning and development, and thus can benefit from out findings. In order to <u>improve infrastructure for research</u>, we also plan to make our tools and raw data available to other researchers. The project will <u>promote teaching, training, and learning</u> by including graduate and undergraduate students in the research project. These students will have the opportunity to develop skills in data collection and analysis.

Results from prior NSF funding

Kevin Crowston has been funded by four NSF grants within the past 48 months. The two grants most closely related to the current proposal (NSF grant IIS 04–14468, *Effective work practices for Open Source Software development*, $327,026, 15 August 2004 to 14 August 2006, and SGER IIS 03–41475, $12,052, 1 September 2003 to 31 August 2004) were discussed above on page 3 in the literature review. These grants have provided support for travel to conferences (e.g., *ApacheCon* and *OSCon*) to observe, interview and seek support from developers and to present preliminary results, and for the purchase of data analysis software and equipment. This work has resulted in one journal paper [37] several conference papers [e.g., 31,34] and workshop presentations [32,33,36,46,96], with additional papers in preparation [35,38,39]. The current grant is sought to continue this research, focusing on issues that have been revealed.

The most recent grant is IIS 04–14482 ($302,685, 1 January 2005 to 31 December 2006), for "How can document-genre metadata improve information-access for large digital collections?" (with Barbara Kwasnik). This project started less than one month ago, so there are no specific results as yet to report, though earlier work by the PIs on genre has appeared in journal [e.g., 41] and conference papers [e.g., 109]. The grant has partially supported work on a conference mini-track and journal special issue [110].

Earlier support came from IIS–0000178 ($269,967, July 1, 2000 to June 30, 2003), entitled *Towards Friction-Free Work: A Multi-Method Study of the Use of Information Technology in the Real Estate Industry*. The goal of that study was to examine how the pervasive use of information and communication technologies (ICT) in the real-estate industry changes the way people and organizations in that industry work. Initial fieldwork resulted in several journal articles [44,48,160] and numerous conference presentations [e.g., 43,47]. The PIs are currently working with the National Association of Realtors to extend and disseminate these results and are planning a follow-on study.

References

1 Abbott, A. (1990) A primer on sequence methods. *Organization Science*, 1(4), 375–392

2 Ahuja, M.K. and Carley, K. (1998) Network structure in virtual organizations. *Journal of Computer-Mediated Communication*, 3(4)

3 Ahuja, M.K., Carley, K. and Galletta, D.F. (1997) Individual performance in distributed design groups: An empirical study. In *SIGCPR Conference*, pp. 160–170, San Francisco, ACM

4 Albino, V., Kuhtz, S. and Scozzi, B. (2003) Actors and cognitive maps on sustainable development in industrial district. In *Uddevalla Symposium*, Uddevalla, Sweden

5 Alho, K. and Sulonen, R. (1998) Supporting virtual software projects on the Web. In *Workshop on Coordinating Distributed Software Development Projects, 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '98)*

6 Arent, J. and Nørbjerg, J. (2000) Software Process Improvement as Organizational Knowledge Creation: A Multiple Case Analysis. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pp. 11 pages, IEEE Press

7 Argyris, C. and Schön, D.A. (1996) *Organizational Learning II: Theory, method and practice*, Reading, MA: Addison-Wesley

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

9 Asklund, U. and Bendix, L. (2001) *Configuration Management for Open Source Software* (R-01-5005) Department of Computer Science, Aalborg University

10 Augustin, L., Bressler, D. and Smith, G. (2002) Accelerating software development through collaboration. In *International Conference on Software Engineering (ICSE)*, pp. 559–563, Orlando, FL

11 Baker-Brown, G., Ballard, E., Bluck, S., DeVries, B., Suedfeld, P. and Tetlock, P. (1990) *Coding Manual for Conceptual/Integrative Complexity* University of British Columbia and University of California, Berkely

12 Bandow, D. (1997) Geographically distributed work groups and IT: A case study of working relationships and IS professionals. In *Proceedings of the SIGCPR Conference*, pp. 87–92

13 Barley, S.R. (1986) Technology as an occasion for structuring: Evidence from the observation of CT scanners and the social order of radiology departments. *Administrative Sciences Quarterly*, 31, 78–109

14 Barley, S.R. and Tolbert, P.S. (1997) Institutionalization and structuration: Studying the links between action and institution. *Organization Studies*, 18(1), 93–117

15 Bessen, J. (2002) *Open Source Software: Free Provision of Complex Public Goods* Research on Innovation

16 Bezroukov, N. (1999) Open source software development as a special type of academic research (critique of vulgar raymondism). *First Monday*, 4(10)

17 Bonneaud, S., Ripoche, G. and Sansonnet, J.-P. (2004) A socio-cognitive model for the characterization of schemes of interaction in distributed collectives. In *Workshop on Distributed Collective Practice: Building new Directions for Infrastructural Studies, CSCW 2004*, Available from: http://www.limsi.fr/Individu/turner/DCP/Chicago2004 /Bonneaud.pdf, Accessed 23 January 2005

18 Brandon, D.P. and Hollingshead, A.B. (2004) Transactive Memory Systems in Organizations: Matching Tasks, Expertise, and People. *Organization Science*, 15(6), 633–644

19 Brooks, F.P., Jr. (1975) *The Mythical Man-month: Essays on Software Engineering*, Reading, MA: Addison-Wesley

20 Brown, J.S. and Duguid, P. (1991) Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organization Science*, 2(1), 40–57

21 Butler, B., Sproull, L., Kiesler, S. and Kraut, R. (2002) Community effort in online groups: Who does the work and why? In *Leadership at a Distance* (Weisband, S. and Atwater, L., eds.), Mahwah, NJ: Lawrence Erlbaum

22 Bélanger, F. and Collins, R. (1998) Distributed Work Arrangements: A Research Framework. *The Information Society*, 14(2), 137–152

23 Cannon-Bowers, J.A. and Salas, E. (1993) Shared mental models in expert decision making. In *Individual and Group Decision Making* (Castellan, N.J., ed.), pp. 221-246, Hillsdale, NJ: Lawrence Erlbaum Associates

24 Cannon-Bowers, J.A. and Salas, E. (2001) Reflections on shared cognition. *Journal of Organizational Behavior*, 22, 195–202

25 Carbonara, N. and Scozzi, B. (2003) Cognitive maps to analyze new product development processes: A case study. In *10th International Product Development Management Conference*, Brussels, Belgium

26 Carley, K.M. (1997) Extracting team mental models through textual analysis. *Journal of Organizational Behaviour*, 18, 533–558

27 Carley, K.M. and Palmquist, M. (1992) Extracting, representing and analyzing mental models. *Social Forces*, 70(3), 601–636

28 Carmel, E. and Agarwal, R. (2001) Tactical approaches for alleviating distance in global software development. *IEEE Software*(March/April), 22–29

29 Cassell, P., ed. (1993) *The Giddens Reader*, Stanford University Press

30 Cox, A. (1998) Cathedrals, Bazaars and the Town Council. Available from: http://slashdot.org/features/98/10/13/1423253.shtml, Accessed 22 March 2004

31 Crowston, K., Annabi, H. and Howison, J. (2003) Defining Open Source Software project success. In *Proceedings of the 24th International Conference on Information Systems (ICIS 2003)*, Seattle, WA:

32 Crowston, K., Annabi, H., Howison, J. and Masango, C. (2004) Effective work practices for Software Engineering: Free/Libre Open Source Software Development. In *WISER Workshop on Interdisciplinary Software Engineering Research, SIGSOFT 2004/FSE-12 Conference*, Newport Beach, CA

33 Crowston, K., Annabi, H., Howison, J. and Masango, C. (2004) Towards a portfolio of FLOSS project success measures. In *Workshop on Open Source Software Engineering, 26th International Conference on Software Engineering*, Edinburgh

34 Crowston, K., Annabi, H., Howison, J. and Masango, C. (2005) Effective work practices for FLOSS development: A model and propositions. In *Proceedings of the Hawai'i International Conference on System Science (HICSS)*, Big Island, Hawai'i:

35 Crowston, K., Heckman, R., Annabi, H. and Masango, C. (Under review) A structurational perspective on leadership in Free/Libre Open Source Software teams.

36 Crowston, K. and Howison, J. (2003) The social structure of Open Source Software development teams. In *The IFIP 8.2 Working Group on Information Systems in Organizations Organizations and Society in Information Systems (OASIS) 2003 Workshop*, Seattle, WA

37 Crowston, K. and Howison, J. (2005) The social structure of free and open source software development. *First Monday*, 10(2)

38 Crowston, K. and Howison, J. (Under review) Hierarchy and Centralization in Free and Open Source Software team communications.

39 Crowston, K., Howison, J., Masango, C. and Eseryel, U.Y. (Under review) Face-to-face interactions in self-organizing distributed teams.

40 Crowston, K. and Kammerer, E. (1998) Coordination and collective mind in software requirements development. *IBM Systems Journal*, 37(2), 227–245

41 Crowston, K. and Kwasnik, B.H. (2003) Can document-genre metadata improve information access to large digital collections? *Library Trends*, 52(2), 345–361

42 Crowston, K. and Osborn, C.S. (2003) A coordination theory approach to process description and redesign. In *Organizing Business Knowledge: The MIT Process Handbook* (Malone, T.W. et al., eds.), Cambridge, MA: MIT Press

43 Crowston, K., Sawyer, S. and Wigand, R. (1999) Investigating the interplay between structure and technology in the real estate industry. In *Organizational Communications and Information Systems Division, Academy of Management Conference*, Chicago, IL

44 Crowston, K., Sawyer, S. and Wigand, R. (2001) Investigating the interplay between structure and technology in the real estate industry. *Information, Technology and People*, 14(2), 163–183

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

46 Crowston, K. and Scozzi, B. (2004) Coordination practices for bug fixing within FLOSS development teams. In *Presentation at 1st International Workshop on Computer Supported Activity Coordination, 6th International Conference on Enterprise Information Systems*, Porto, Portugal

47 Crowston, K. and Wigand, R. (1998) Use of the web for electronic commerce in real estate. In *Association for Information Systems Americas Conference*, Baltimore, MD

48 Crowston, K. and Wigand, R. (1999) Real estate war in cyberspace: An emerging electronic market? *International Journal of Electronic Markets*, 9(1–2), 1–8

49 Cubranic, D. and Booth, K.S. (1999) Coordinating Open Source Software development. In *Proceedings of the 7th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*

50 Cummings, J.N. and Cross, R. (2003) Structural properties of work groups and their consequences for performance. *Social Networks*, 25, 197–210

51 Curtis, B., Krasner, H. and Iscoe, N. (1988) A field study of the software design process for large systems. *CACM*, 31(11), 1268–1287

52 Curtis, B., Walz, D. and Elam, J.J. (1990) Studying the process of software design teams. In *Proceedings of the 5th International Software Process Workshop On Experience With Software Process Models*, pp. 52–53, Kennebunkport, Maine, United States:

53 de Souza, P.S. (1993) *Asynchronous Organizations for Multi-Algorithm Problems*, Doctoral Thesis, Department of Electrical and Computer Engineering, Carnegie-Mellon University

54 DeSanctis, G. and Poole, M.S. (1994) Capturing the complexity in advanced technology use: Adaptive structuration theory. *Organization Science*, 5(2), 121–147

55 Di Bona, C., Ockman, S. and Stone, M., eds (1999) *Open Sources: Voices from the Open Source Revolution*, O'Reilly & Associates

56 Dougherty, D. (1992) Interpretive barriers to successful product innovation in large firms. *Organization Science*, 3(2), 179–202

57 Ducheneaut, N. (2003) *The reproduction of Open Source Software programming communities*, PhD Thesis, Information Management and Systems, University of California, Berkeley

58 Dutoit, A.H. and Bruegge, B. (1998) Communication Metrics for Software Development. *IEEE Transactions On Software Engineering*, 24(8), 615–628

59 Edmondson, A. (1999) Psychological safety and learning behavior in work teams. *Administrative Science Quarterly*, 44(2), 350-383

60 Edwards, K. (2001) Epistemic communities, situated learning and Open Source Software development. In *Epistemic Cultures and the Practice of Interdisciplinarity Workshop*, NTNU, Trondheim

61 Ellis, A.P.J., Hollenbeck, J.R., Ilgen, D.R., Porter, C.O.L.H., West, B.J. and Moon, H. (2003) Team learning: Collectively connecting the dots. *Journal of Applied Psychology*, 88(5), 821-835

62 Espinosa, J.A., Kraut, R.E., Lerch, J.F., Slaughter, S.A., Herbsleb, J.D. and Mockus, A. (2001) Shared mental models and coordination in large-scale, distributed software development. In *Twenty-Second International Conference on Information Systems*, pp. 513–518, New Orleans, LA

63 Espinosa, J.A., Lerch, F.J. and Kraut, R.E. (2004) Explicit versus implicit coordination mechanisms and task dependencies: One size does not fit all. In *Team cognition: Understanding the factors that drive process and performance* (Salas, E. and Fiore, S.M., eds.), pp. 107-129, Washington, DC: APA

64 Feller, J. (2001) Thoughts on Studying Open Source Software Communities. In *Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective* (Russo, N.L. et al., eds.), pp. 379–388Kluwer

65 Fielding, R.T. (1997) The Apache Group: A case study of Internet collaboration and virtual communities. Available from: http://www.ics.uci.edu/fielding/talks/ssapache /overview.htm.

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

67 Finholt, T. and Sproull, L.S. (1990) Electronic groups at work. *Organization Science*, 1(1), 41–64

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

69 Gallivan, M.J. (2001) Striking a balance between trust and control in a virtual organization: A content analysis of open source software case studies. *Information Systems Journal*, 11(4), 277–304

70 Garvin, D.A. (1991) Barriers and gateways to learning. In *Education for Judgement: The Art of Discussion Leadership* (Christensen, C.R., Garvin, D.A. & Sweet, A., ed.), pp. 3–14, Boston: Harvard Business School Press

71 Gasser, L. and Ripoche, G. (2003) Distributed Collective Practices and F/OSS Problem Management: Perspective and Methods. In *Conference on Cooperation, Innovation & Technologie (CITE2003)*, University de Technologie de Troyes, France, Available from: http://www.ics.uci.edu/~wscacchi/Papers/UIUC/gasser-ripoche-cite.pdf, Accessed 21 January 2005

72 Gasser, L., Scacchi, W., Ripoche, G. and Penne, B. (2003) Understanding Continuous Design in F/OSS Projects. In *16th International Conference on Software Engineering &*

its Applications (ICSSEA-03), Paris, Frane, Available from: http://www.ics.uci.edu /~wscacchi/Papers/New/ICSSEA03.pdf, Accessed 21 January 2005

73 German, D.M. (2002) The evolution of the GNOME Project. In *Meeting Challenges and Surviving Success: 2nd ICSE Workshop on Open Source Software Engineering*, Orlando, FL, Available from: http://opensource.ucc.ie/icse2002/German.pdf, Accessed 23 January 2005

74 Giddens, A. (1984) *The Constitution of Society: Outline of the Theory of Structuration*, Berkeley: University of California

75 Giuri, P., Ploner, M., Rullani, F. and Torrisi, S. (2004) *Skills and openness of OSS projects: Implications for performance*, Working paper Laboratory of Economics and Management, Sant'Anna School of Advanced Studies, Available from: http://www.lem.sssup.it/WPLem/files/2004-19.pdf, Accessed 21 January 2005

76 González-Barahona, J.M. and Robles, G. (2003) Free Software Engineering: A Field to Explore. *Upgrade*, 4(4), 49–54

77 Grabowski, M. and Roberts, K.H. (1999) Risk mitigation in virtual organizations. *Organization Science*, 10(6), 704–721

78 Grant, R.M. (1996) Toward a knowledge-based theory of the firm. *Strategic Management Journal*, 17(Winter), 109–122

79 Gregory, D. (1989) Presences and absences: Time-space relations and structuration theory. In *Social Theory of Modern Societies: Anthony Giddens and His Critics*, Cambridge: Cambridge University Press

80 Griffith, T. and Neale, M.A. (1999) *Information Processing and Performance in Traditional and Virtual Teams: The Role of Transactive Memory*, Research Paper (1613) Stanford University Graduate School of Business, Available from: http://www.gsb.stanford.edu/cebc/pdfs/rp1611.pdf, Accessed 20 January 2005

81 Guzzo, R.A. and Dickson, M.W. (1996) Teams in organizations: Recent research on performance effectiveness. *Annual Review of Psychology*, 47, 307–338

82 Hackman, J.R. (1987) The design of work teams. In *The Handbook of Organizational Behavior* (Lorsch, J.W., ed.), pp. 315–342, Englewood Cliffs, NJ: Prentice-Hall

83 Hallen, J., Hammarqvist, A., Juhlin, F. and Chrigstrom, A. (1999) Linux in the workplace. *IEEE Software*, 16(1), 52–57

84 Halloran, T.J. and Scherlis, W.L. (2002) High Quality and Open Source Software Practices. In *Meeting Challenges and Surviving Success: 2nd ICSE Workshop on Open Source Software Engineering*, Orlando, FL, Available from: http://www.fluid.cs.cmu.edu:8080/Fluid/fluid-publications/HalloranScherlis.pdf, Accessed 21 January 2005

85 Hann, I.-H., Roberts, J., Slaughter, S. and Fielding, R. (2002) Economic incentives for participating in open source software projects. In *Proceedings of the Twenty-Third International Conference on Information Systems*, pp. 365–372

86 Hann, I.-H., Roberts, J. and Slaughter, S.A. (2004) Why developers participate in open source software projects: An empirical investigation. In *Twenty-Fifth International Conference on Information Systems*, pp. 821–830, Washington, DC

87 Hare, A.P. (1976) *Handbook of Small Group Research*, New York: Free Press

88 Hayes, J. and Allinson, C.W. (1998) Cognitive style and the theory and practice of individual and collective learning in organizations. *Human Relations*, 51(7), 847-871

89 Hecker, F. (1999) Mozilla at one: A look back and ahead. Available from: http://www.mozilla.org/mozilla-at-one.html

90 Heckman, R. and Annabi, H. (2003) A content analytic comparison of FTF and ALN case-study discussions. In *36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, Big Island, Hawaii, IEEE Press, Available from: http://csdl.computer.org/comp/proceedings/hicss/2003/1874/01/187410003aabs.htm

91 Hemetsberger, A. and Reinhardt, C. (2004) Sharing and Creating Knowledge in Open-Source Communities: The case of KDE. In *The Fifth European Conference on Organizational Knowledge, Learning, and Capabilities*, Innsbruck, Austria

92 Herbsleb, J.D. and Grinter, R.E. (1999) Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software*(September/October), 63–70

93 Herbsleb, J.D. and Grinter, R.E. (1999) Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of the International Conference on Software Engineering (ICSE '99)*, pp. 85–95, Los Angeles, CA: ACM

94 Herbsleb, J.D., Mockus, A., Finholt, T.A. and Grinter, R.E. (2001) An empirical study of global software development: Distance and speed. In *Proceedings of the International Conference on Software Engineering (ICSE 2001)*, pp. 81–90, Toronto, Canada:

95 Hertel, G., Niedner, S. and Herrmann, S. (n.d.) *Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel* University of Kiel

96 Howison, J. and Crowston, K. (2004) The perils and pitfalls of mining SourceForge. In *Presentation at the Workshop on Mining Software Repositories, 26th International Conference on Software Engineering*, Edinburgh, Scotland

97 Huber, G.P. (1991) Organizational learning: The contributing processes and the literatures. *Organization Science*, 2(1), 88–115

98 Humphrey, W.S. (2000) *Introduction to Team Software Process*: Addison-Wesley

99 Jarvenpaa, S.L. and Leidner, D.E. (1999) Communication and trust in global virtual teams. *Organization Science*, 10(6), 791–815

100 Jensen, C. and Scacchi, W. (2005) Collaboration, Leadership, Control, and Conflict Negotiation in the Netbeans.org Open Source Software Development Community. In *Proceedings of the Hawai'i International Conference on System Science (HICSS)*, Big Island, Hawai'i:

101 Jørgensen, N. (2001) Putting it all in the trunk: incremental software development in the FreeBSD open source project. *Information Systems Journal*, 11(4), 321–336

102 Kiesler, S. and Cummings, J. (2002) What do we know about proximity and distance in work groups? A legacy of research. In *Distributed Work* (Hinds, P. and Kiesler, S., eds.), pp. 57–80, Cambridge, MA: MIT Press

103 Koch, S. and Schneider, G. (2002) Effort, co-operation and co-ordination in an open source software project: GNOME. *Information Systems Journal*, 12(1), 27–42

104 Kogut, B. and Metiu, A. (2001) Open-source software development and distributed innovation. *Oxford Review of Economic Policy*, 17(2), 248–264

105 Kraut, R.E., Steinfield, C., Chan, A.P., Butler, B. and Hoag, A. (1999) Coordination and virtualization: The role of electronic networks and personal relationships. *Organization Science*, 10(6), 722–740

106 Kraut, R.E. and Streeter, L.A. (1995) Coordination in software development. *Communications of the ACM*, 38(3), 69–81

107 Krishnamurthy, S. (2002) *Cave or Community? An Empirical Examination of 100 Mature Open Source Projects* University of Washington, Bothell

108 Kuwabara, K. (2000) Linux: A bazaar at the edge of chaos. *First Monday*, 5(3)

109 Kwasnik, B.H. and Crowston, K. (2004) A framework for creating a facetted classification for genres: Addressing issues of multidimensionality. In *Proceedings of the Hawai'i International Conference on System Science (HICSS)*, Big Island, Hawai'i:

110 Kwaśnik, B.H. and Crowston, K. (In press) Genres of digital documents: Introduction to the special issue. *Information, Technology & People*

111 Langfield-Smith, K. (1992) Exploring the need for a shared cognitive map. *Journal of management studies*, 29(3), 349-368

112 Lanzara, G.F. and Morner, M. (2003) The Knowledge Ecology of Open-Source Software Projects. In *19th EGOS Colloquium*, Copenhagen

113 Lanzara, G.F. and Morner, M. (2004) Making and sharing knowledge at electronic crossroads: the evolutionary ecology of open source. In *5th European Conference on Organizational Knowledge, Learning and Capabilities*, Innsbruck, Austria

114 Lee, G.K. and Cole, R.E. (2003) From a firm-based to a community-based model of knowledge creation: The case of Linux kernel development. *Organization Science*, 14(6), 633–649

115 Leibovitch, E. (1999) The business case for Linux. *IEEE Software*, 16(1), 40–44

116 Lerner, J. and Tirole, J. (2001) The open source movement: Key research questions. *European Economic Review*, 45, 819–826

117 Levesque, L.L., Wilson, J.M. and Wholey, D.R. (2001) Cognitive divergence and shared mental models in software development project teams. *Journal of Organization Behavior*, 22, 135–144

118 Lin, Y. (2004) Epistemologically Multiple Actor-Centred System: or, EMACS at work! In *3rd Oekonux Conference*, Vienna, Austria

119 Ljungberg, J. (2000) Open Source Movements as a Model for Organizing. *European Journal of Information Systems*, 9(4)

120 Madey, G., Freeh, V. and Tynan, R. (2002) The Open Source Software development phenomenon: An analysis based on social network theory. In *Proceedings of the Eighth Americas Conference on Information Systems*, pp. 1806–1815

121 Maier, G.W., Prange, C. and Rosenstiel, L. (2001) Psychological perspectives on organizational learning. In *Handbook of Organizational Learning and Knowledge* (Dierkes, M. et al., eds.), pp. 14–34, New York: Oxford Press

122 Majchrzak, A. and Malhotra, A. (2004) *Virtual Workspace Technology Use and Knowledge-Sharing Effectiveness in Distributed Teams: The Influence of a Team's Transactive Memory* Marshall School of Business, University of Southern California, Available from: http://oz.stern.nyu.edu/seminar/0928.pdf, Accessed 23 January 2004

123 March, J.G., Schulz, M. and Zhou, X. (2000) *The Dynamics of Rules: Change in Written Organizational Codes*, Stanford, CA: Stanford University Press

124 Mark, G. (2002) Conventions for coordinating electronic distributed work: A longitudinal study of groupware use. In *Distributed Work* (Hinds, P. and Kiesler, S., eds.), pp. 259–282, Cambridge, MA: MIT Press

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

126 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

127 Miles, M.B. and Huberman, A.M. (1994) *Qualitative Data Analysis: An Expanded Sourcebook*, Thousand Oaks: Sage Publications

128 Miner, A.S. and Mezias, S.J. (1996) Ugly Duckling No More: Pasts and Futures of Organizational learning. *Organization Science*, 7(1), 88–99

129 Mockus, A., Fielding, R.T. and Herbsleb, J.D. (2000) A case study of Open Source Software development: The Apache server. In *Proceedings of ICSE'2000*, pp. 11 pages

130 Mockus, A., Fielding, R.T. and Herbsleb, J.D. (2002) Two Case Studies Of Open Source Software Development: Apache And Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3), 309–346

131 Mohammed, S. and Dumville, B.C. (2001) Team mental models in a team knowledge framework: Expanding theory and measurement across disciplinary boundaries. *Journal of Organizational Behavior*, 22(2), 89–106

132 Moody, G. (2001) *Rebel code—Inside Linux and the open source movement*, Cambridge, MA: Perseus Publishing

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

134 Mortensen, M. and Hinds, P. (2002) Fuzzy teams: Boundary disagreement in distributed and collocated teams. In *Distributed Work* (Hinds, P. and Kiesler, S., eds.), pp. 284–308, Cambridge, MA: MIT Press

135 Nadkarni, S. and Nah, F.F.-H. (2003) Aggregated causal maps: An approach to elicit and aggregate the knowledge of multiple experts. *Communications of the Association for Information Systems*, 12, 406–436

136 Nardi, B.A. and Whittaker, S. (2002) The place of face to face communication in distributed work. In *Distributed Work: New Research on Working across Distance Using Technology* (Hinds, P. and Kiesler, S., eds.), pp. 83–110, Cambridge, MA: MIT Press

137 Nejmeh, B.A. (1994) Internet: A strategic tool for the software enterprise. *Communications of the ACM*, 37(11), 23–27

138 Newman, M. and Robey, D. (1992) A social process model of user-analyst relationships. *MIS Quarterly*, 16(2), 249–266

139 O'Leary, M., Orlikowski, W.J. and Yates, J. (2002) Distributed work over the centuries: Trust and control in the Hudson's Bay Company, 1670–1826. In *Distributed Work* (Hinds, P. and Kiesler, S., eds.), pp. 27–54, Cambridge, MA: MIT Press

140 O'Mahony, S. and Ferraro, F. (2003) Managing the Boundary of an 'Open' Project. In *Santa Fe Institute (SFI) Workshop on The Network Construction of Markets* (Padgett, J. and Powell, W., eds.), Available from: http://opensource.mit.edu/papers /omahonyferraro.pdf

141 Ocker, R.J. and Fjermestad, J. (2000) High versus low performing virtual design teams: A preliminary analysis of communication. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pp. 10 pages

142 Orlikowski, W.J. (1992) The duality of technology: Rethinking the concept of technology in organizations. *Organization Science*, 3(3), 398–427

143 Orlikowski, W.J. (2000) Using technology and constituting structures: A practice lens for studying technology in organizations. *Organization Science*, 11(4), 404–428

144 Orlikowski, W.J. (2002) Knowing in practice: Enacting a collective capability in distributed organizing. *Organization Science*, 13(3), 249–273

145 Orlikowski, W.J. and Yates, J. (1994) Genre repertoire: The structuring of communicative practices in organizations. *Administrative Sciences Quarterly*, 33, 541–574

146 Orr, J. (1996) *Talking About Machines: An Ethnography of a Modern Job*, Ithaca, NY: ILR Press

147 O'Leary, M. and Cummings, J. (2002) The Spatial, Temporal, and Configurational Characteristics of Geographic Dispersion in Teams. In *Academy of Management Conference*, Denver, CO

148 O'Reilly, T. (1999) Lessons from open source software development. *Communications of the ACM*, 42(4), 33–37

149 Pfaff, B. (1998) Society and open source: Why open source software is better for society than proprietary closed source software. Available from: http://www.msu.edu/user /pfaffben/writings/anp/oss-is-better.html

150 Prasad, G.C. (n.d.) A hard look at Linux's claimed strengths…. Available from: http://www.osopinion.com/Opinions/GaneshCPrasad/GaneshCPrasad2-2.html

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

152 Raymond, E.S. (1998) Homesteading the noosphere.

153 Rentsch, J.R. and Klimonski, R.J. (2001) Why do 'great minds' think alike? Antecedents of team member schema agreement. *Journal of Organizational Behavior*, 22(2), 107–120

154 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

155 Rossi, M.A. (2004) *Decoding the "Free/Open Source (F/OSS) Software Puzzle": A survey of theoretical and empirical contributions*, Working paper (424) Università degli Studi di Siena, Dipartimento Di Economia Politica

156 Sagers, G.W. (2004) The influence of network governance factors on success in open source software development projects. In *Twenty-Fifth International Conference on Information Systems*, pp. 427–438, Washington, DC

157 Sagers, G.W., Wasko, M.M. and Dickey, M.H. (2004) Coordinating Efforts in Virtual Communities: Examining Network Governance in Open Source. In *Tenth Americas Conference on Information Systems*, pp. 2695–2698, New York, NY

158 Sarason, Y. (1995) A model of organizational transformation: The incorporation of organizational identity into a structuration theory framework. *Academy of Management Journal*(Best papers proceedings), 47–51

159 Sawyer, S. (2000) A Social Analysis of Software Development Teams: Three Models and their Differences. In *The 2000 Americas Conference on Information Systems (AMCIS 2000)*, pp. 1645–1649

160 Sawyer, S., Crowston, K., Wigand, R. and Allbritton, M. (2003) The social embeddedness of transactions: Evidence from the residential real estate industry. *The Information Society*, 19(2), 135–154

161 Sawyer, S. and Guinan, P.J. (1998) Software development: Processes and performance. *IBM Systems Journal*, 37(4), 552–568

162 Scacchi, W. (1991) The software infrastructure for a distributed software factory. *Software Engineering Journal*, 6(5), 355–369

163 Scacchi, W. (2002) Understanding the requirements for developing Open Source Software systems. *IEE Proceedings Software*, 149(1), 24–39

164 Scacchi, W. (2004) Free/Open Source Software Development Practices in the Computer Game Community. *IEEE Software*, 21(1), 56–66

165 Seaman, C.B. and Basili, V.R. (1997) *Communication and Organization in Software Development: An Empirical Study* Institute for Advanced Computer Studies, University of Maryland

166 Shepard, T., Lamb, M. and Kelly, D. (2001) More testing should be taught. *Communication of the ACM*, 44(6), 103–108

167 Stamelos, I., Angelis, L., Oikonomou, A. and Bleris, G.L. (2002) Code quality analysis in open source software development. *Information Systems Journal*, 12(1), 43–60

168 Stein, E.W. and Vandenbosch, B. (1996) Organizational learning during advanced system development: Opportunities and obstacles. *Journal of Management Information Systems*, 13(2), 115–136

169 Stewart, K.J. and Ammeter, T. (2002) An exploratory study of factors influencing the level of vitality and popularity of open source projects. In *Proceedings of the Twenty-Third International Conference on Information Systems*, pp. 853–857

170 Stewart, K.J. and Gosain, S. (2001) Impacts of ideology, trust, and communication on effectivness in open source software development teams. In *Twenty-Second International Conference on Information Systems*, pp. 507–512, New Orleans, LA

171 Sutanto, J., Kankanhalli, A. and Tan, B.C.Y. (2004) Task coordination in global virtual teams. In *Twenty-Fifth International Conference on Information Systems*, pp. 807–820, Washington, DC

172 Swieringa, J. and Wierdsma, A. (1992) *Becoming a Learning Organization*, Reading, MA: Addison-Wesley

173 Turner, W., Sansonnet, J.-P., Gasser, L. and Ripoche, G. (2004) Confidence-based organizational metrics. In *Workshop on Distributed Collective Practice: Building new Directions for Infrastructural Studies, CSCW 2004*, Available from: http://www.limsi.fr /Individu/turner/DCP/Chicago2004/Turner.pdf, Accessed 23 January 2005

174 Valloppillil, V. (1998) Halloween I: Open Source Software. Available from: http://www.opensource.org/halloween/halloween1.html

175 Valloppillil, V. and Cohen, J. (1998) Halloween II: Linux OS Competitive Analysis. Available from: http://www.opensource.org/halloween/halloween2.html

176 van de Ven, A.H. and Poole, M.S. (1990) Methods for studying innovation development in the Minnesota Innovations Research Program. *Organization Science*, 1(3), 313–335

177 van Fenema, P.C. (2002) *Coordination and control of globally distributed software projects*, Doctoral Dissertation, Erasmus Research Institute of Management, Erasmus University

178 Vixie, P. (1999) Software engineering. In *Open sources: Voices from the open source revolution* (Di Bona, C. et al., eds.), San Francisco: O'Reilly

179 von Hippel, E. (2001) Innovation by user communities: Learning from open-source software. *Sloan Management Review*(Summer), 82–86

180 von Hippel, E. and von Krogh, G. (2002) *Exploring the Open Source Software Phenomenon: Issues for Organization Science* Sloan School of Management, MIT

181 von Hippel, E. and von Krogh, G. (2003) Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science*, 14(2), 209–213

182 von Krogh, G., Spaeth, S. and Lakhani, K.R. (2003) Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research Policy*, 32(7), 1217–1241

183 Walsham, G. (1993) *Interpreting Information Systems in Organizations*, Chichester: John-Wiley

184 Walton, R.E. and Hackman, J.R. (1986) Groups under contrasting management strategies. In *Designing Effective Work Groups* (Goodman, P.S. and Associates, eds.), pp. 168–201, San Francisco, CA: Jossey-Bass

185 Walz, D.B., Elam, J.J. and Curtis, B. (1993) Inside a software design team: knowledge acquisition, sharing, and integration. *Communications of the ACM*, 36(10), 63–77

186 Watson-Manheim, M.B., Chudoba, K.M. and Crowston, K. (2002) Discontinuities and continuities: A new way to understand virtual work. *Information, Technology and People*, 15(3), 191–209

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

188 Webb, E. and Weick, K.E. (1979) Unobtrusive measures in organizational theory: A reminder. *Administrative Science Quarterly*, 24(4), 650–659

189 Weick, K.E. and Roberts, K. (1993) Collective mind in organizations: Heedful interrelating on flight decks. *Administrative Science Quarterly*, 38(3), 357–381

190 Weisband, S. (2002) Maintaining awareness in distributed team collaboration: Implications for leadership and performance. In *Distributed Work* (Hinds, P. and Kiesler, S., eds.), pp. 311–333, Cambridge, MA: MIT Press

191 Yamauchi, Y., Yokozawa, M., Shinohara, T. and Ishida, T. (2000) Collaboration with lean media: How open-source software succeeds. In *Proceedings of CSCW'00*, pp. 329–338, Philadelphia, PA:

192 Yoo, Y. and Kanawattanachai, P. (2001) Developments of transactive memory systems and collective mind in virtual teams. *International Journal of Organizational Analysis*, 9(2), 187–208