

Investigating the Dynamics of Open Source Software Development Teams

Research on human and social behavior is increasingly characterized by a focus on dynamics—on the evolution of formal and informal organizations over time. We propose a social science study in the context of distributed teams of Free/Libre Open Source (FLOSS) software developers to better understand the cognitive and social structures that underlie changes in individual and team behaviours in these teams. Our study addresses the general research question: What are the dynamics through which distributed teams develop and work?

Increasingly, organizational work is performed by distributed teams of interdependent knowledge workers. These teams have many benefits, but the distance between members creates challenges for 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 developing distributed 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. 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 dynamics 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 dynamics. We will also use social network analysis to study the socialization process of members and change in roles over time.

Expected intellectual contributions

The study will have conceptual, methodological as well as practical contributions. Developing an integrated theoretical framework to understand the dynamics of a distributed team will be a contribution to the study of distributed teams. Understanding the dynamics 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 evolve 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 such an organizational form. 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.

TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.C.

	Total No. of Pages	Page No.* (Optional)*
Cover Sheet for Proposal to the National Science Foundation		
Project Summary (not to exceed 1 page)	1	_____
Table of Contents	1	_____
Project Description (Including Results from Prior NSF Support) (not to exceed 15 pages) (Exceed only if allowed by a specific program announcement/solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)	15	_____
References Cited	7	_____
Biographical Sketches (Not to exceed 2 pages each)	2	_____
Budget (Plus up to 3 pages of budget justification)	5	_____
Current and Pending Support	1	_____
Facilities, Equipment and Other Resources	1	_____
Special Information/Supplementary Documentation	15	_____
Appendix (List below.) (Include only if allowed by a specific program announcement/ solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)	_____	_____
Appendix Items:		

*Proposers may select any numbering mechanism for the proposal. The entire proposal however, must be paginated. Complete both columns only if the proposal is numbered consecutively.

Investigating the Dynamics of Open Source Software Development Teams

Research on human and social behavior is increasingly characterized by a focus on dynamics—on the evolution of formal and informal organizations over time. We propose a social science study in the context of distributed teams of software developers to better understand the cognitive and social structures that underlie changes in individual and team behaviours in these teams. We will address the general research questions:

What are the dynamics through which members of distributed ICT-supported teams of Open Source Software developers form shared mental models, individual roles, informal norms and formal rules and how do these structures guide their behaviours?

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.

Revolutionary technologies and ideas have created a more closely linked world with almost instantaneous transmission of information to feed a global economy. A prominent example of this transformation is the emergence of Free/Libre Open Source Software (FLOSS, e.g., Linux or Apache), created by distributed dynamic teams of volunteers and professionals working in 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 (“free as in beer”). Much (though not all) of this software is also “free software”, meaning that derivative works must be made available under the same unrestricted license terms (“free as in speech”, thus “libre”). We have chosen to use the acronym FLOSS rather than the more common OSS to emphasize this dual meaning. 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 [2].

Key to our interest is the fact that most FLOSS software is developed by distributed teams. 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) [3,4]. These teams depend on processes that span traditional boundaries of place and ownership. The research literature on software development and on distributed work emphasizes the difficulties of distributed software development, but the case of 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 [5]. As well, many (though by no means all) programmers contribute to projects as volunteers, without working for a common organization or being paid. Characterized by a globally distributed developer force and a rapid and reliable software development process, effective FLOSS development teams somehow profit from the advantages and overcome the challenges of distributed work [6]. 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. 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 [7] 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”. As evidenced by the attached letters of support from FLOSS developers,

members of the FLOSS community are themselves interested in understanding and documenting effective practices and teams so as to improve their performance.

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 section 2, we develop a conceptual framework for our study, drawing on theories of shared mental models [8,9] and organizational learning [10,11], and using structuration theory [1] as an organizing framework. In section 3, we present the study design, with details of the data collection and analysis plans. We conclude 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., 12], advances in information and communication technologies have been crucial enablers for recent developments of this organizational form [13]. Distributed teams seem particularly attractive for software development because the code can be shared via the systems used to support team interactions [14,15]. While distributed teams have many potential benefits, distributed workers face many real challenges. Watson-Manheim, Chudoba, & Crowston [16] 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 [17], or that produces unintended information filtering [18] or misunderstandings [19]. These interpretative difficulties in turn make it hard for team members to develop shared mental models of the developing project [20,21]. A lack of common knowledge about the status, authority and competencies of team participants can be an obstacle to the development of team norms [22] and conventions [23].

The presence of discontinuities seems likely to be particularly problematic for software developers [17]. Numerous studies of the social aspects of software development teams [17,24-27] conclude that large system development requires knowledge from many domains, which is thinly spread among different developers [24]. As a result, large projects require a high degree of knowledge integration and the coordinated efforts of multiple developers [28]. More effort is required for interaction when participants are distant and unfamiliar with each others work [29,30]. The additional effort required for distributed work often translates into delays in software release compared to traditional face-to-face teams [31,32]. 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 [5].

In response to the problems created by discontinuities, studies of distributed teams stress the need for a significant amount of time spent learning how to communicate, interact and socialize using computer-supported communications tools [33]. Research has shown the importance of formal and informal coordination mechanisms and information sharing [25] 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 [34-38]. Successful distributed teams share knowledge and information and create new practices to meet the task and social needs of the members [39]. However, the dynamics of knowledge sharing and socialization for distributed teams are still open topics for research [e.g., 40].

Research on FLOSS development

The nascent research literature on FLOSS has addressed a variety of questions. First, 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., 41,42,43]. Second, various explanations have been proposed for the decision by individuals to contribute to projects without pay [e.g., 44,45-48]. These authors have mentioned factors such as personal interest,

ideological commitment, development of skills [49] or enhancement of reputation [48]. Finally, a few authors have investigated the processes of FLOSS development [e.g., 3,50], which is the focus of this proposal.

Raymond's [3] bazaar metaphor is perhaps the most well-known model of the FLOSS process. 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. While popular, the bazaar metaphor has been broadly criticized. According to its detractors, the bazaar metaphor disregards important aspects of the FLOSS process, such as the importance of project leader control, the existence of de-facto hierarchies, the danger of information overload and burnout, and the possibility of conflicts that cause a loss of interest in a project or forking [51,52]. Recent empirical work has begun to illuminate the structure and function of FLOSS development teams. For example, Gallivan [53] analyzes descriptions of the FLOSS process and suggests that teams rely on a variety of social control mechanisms rather than on trust.

The other major stream of research examines factors for the success of FLOSS in general (though there have been few systematic comparison across multiple projects, e.g., [54]). 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 [55-61]. In turn, the quality of the software and speed of development have been attributed to two factors: that developers are also users of the software and the availability of source code. First, FLOSS projects often originate from a personal need [62,63], 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 [64]. (Of course, over-reliance on this mode of requirements gathering may also limit the applicability of the FLOSS model.) Second, 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. Active users also play an important role [65]. 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 [66]. The FLOSS process enables hundreds of people to work on these parts of the process [67].

The studies of FLOSS teams and of distributed teams more generally point to the need to understand the dynamics of their work. In their study of distributed cross-functional teams, Robey et al. [39] 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 [10,68]. The better an organization is at learning, the better it can be at adapting to the environment, correcting for error, and innovating [69]. 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 [39,40]. Thus it is important for us to first understand the dynamics of these teams and of 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" [10].

2. Conceptual development

In this section we develop the conceptual framework for our study. 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 aspects of FLOSS projects suggest analyzing them as work teams. Guzzo and Dickson [70] defined a work team as “made up of 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 co-workers)”.

A structural perspective on team dynamics

To conceptualize the dynamics of these teams and the process of changes within them, we adopt a structural perspective. Numerous authors have used a structural perspective to support empirical analyses of group changes [71-75]. A discussion of the merits of each use is beyond the scope of this application. Here, we build on the view of structuration presented by Orlikowski [72] and Barley and Tolbert [1].

Structuration theory [76] is a broad sociological theory that seeks to unite action and structure and to explain the dynamic of their evolution. 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 [77] 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 [78]. Barley and Tolbert [1] note that structuration is “a continuous process whose operations can be observed only through time” (p. 100). Figure 1, adapted from [1] shows the relation between institution (which the authors use synonymously 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 Cas-sell [79] 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

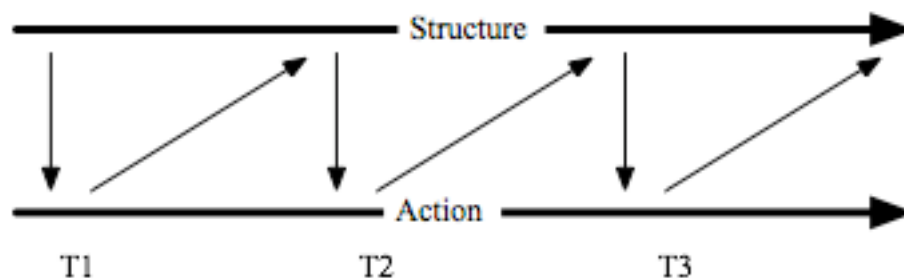


Figure 1. A sequential model of the relation between structure and action [from 1].

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 [78].

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 the figure). Barley and Tolbert [1] note that “actors are more likely to replicate scripted behaviours” than to develop new ones. Orlikowski and Yates [80] 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 particularly 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. 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 [81] model of group performance suggests organizational context as an important factor affecting team processes. Finholt and Sproull [82] 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 effect on team processes and activities.

Conceptualizing structuration in FLOSS teams

To apply structuration as a perspective to conceptualize the dynamics of 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” [1] in the form of interpretive schemes, resources, and norms [1,83]. In the remainder of this section, we elaborate each of these three aspects of structure as they apply to FLOSS development in particular.

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. [84], “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). Without shared mental models, individuals from different teams or backgrounds may interpret tasks differently based on their backgrounds, making collaboration and communication difficult [85]. 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 distributed software development, as in the case of FLOSS teams [86]. Curtis et al. [20], 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 be particularly affect FLOSS development, since FLOSS team members are distributed, have diverse backgrounds, and join in different phases of the software development process. In short, shared mental models are important as guides to effective individual contributions to, and coordination of the software development process.

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

[87] 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 [88], who identify the importance of socialization, conversation and recapitulation. First, new members joining a team need to be socialized into the team to understand how they fit into the process being performed. They need to be encouraged and educated to interact with one another to develop a strong sense of “how we do things around here” (e.g., norms). Barley and Tolbert [1] similarly note that socialization frequently “involves an individual internalizing rules and interpretations of behaviour appropriate for particular settings” (p. 100). Second, 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. 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. Finally, Brown and Duguid [88] 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.

Most studies on shared mental models remain conceptual [89]. The few empirical studies [e.g., 86,90] investigated the relationship between team or organizational factors and the presence of shared mental models. This study will investigate the process through which distributed teams develop shared mental models. This will be accomplished through the analysis of interaction data for evidence of conversations, recapitulation of implicit and explicit rules and ideas about task, team members, attitudes, and beliefs.

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 (<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 role of these sorts of 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.

Several authors have described FLOSS teams as having a hierarchical or onion-like structure [91-93], as shown in Figure 2. At the centre are the core developers, who contribute most of the code and oversee the design and evolution of the project. Core developers are distinguished by having write privileges on the source code. The core is usually small and exhibits a high level of interaction, which would be difficult to maintain if the core group were large. Surrounding the core are 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. 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 from the core are the passive users.

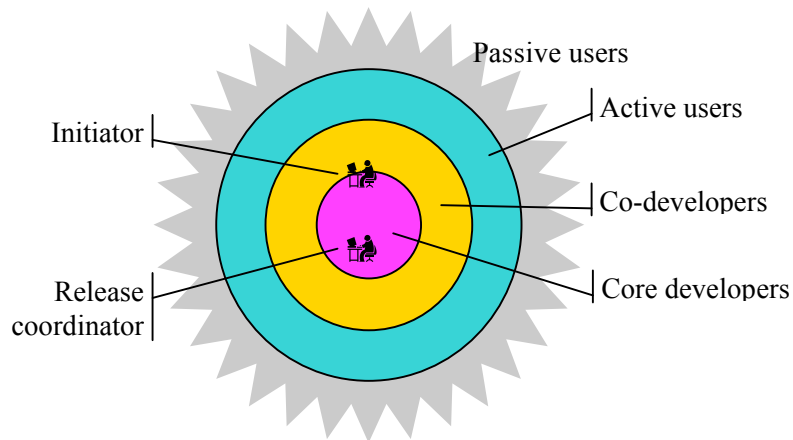


Figure 2. Hypothesized FLOSS development team structure.

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 from the core are the passive users.

The border of the outer circle is indistinct because the nature and variety of FLOSS distribution channels makes it difficult or impossible to know the exact size of the user population.

As their involvement with a project changes, individuals may move from role to role. For example, a common pattern is for active users to be invited 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, while others 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 [94,95]. This barrier is particularly troubling because of the reliance of FLOSS projects on volunteer submission and “fresh blood” [96]. These characteristics again emphasize the importance of socialization and movement of individuals through roles in the projects.

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 [87], describes one aspect of team functions as the creation of rules, implicit and explicit. To conceptualize this aspect of teams, we also draw on Swieringa and Wierdsma’s [97] description of organizations as collections of implicit and explicit rules that guide member behaviours. Implicit rules are team norms, shared amongst members of the team. Explicit rules are the stated rules, policies, procedures and team requirements defined for the team. We are particularly interested in the way these rules guide individual contributions to the team’s goals.

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. [98] and Hayes and Allinson [99] refer to as learning on the group level. Grant [100] 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.

Summary

Combining the discussion of the three aspects of structure described above results in the conceptual framework shown in Table 1. For each of the three aspects of structure, the table describes the embodiment of the structure as we have conceptualized it for FLOSS teams, and the actions that are guided by the structures and that reinforce or modify the structures. The resulting model is largely consistent with Grant’s knowledge-based view of the firm [100], which analyzes a firm as a structure for integrating specialist knowledge into the firm’s activities and products [101]. Though this theory was originally stated in terms of firms, it is easily applicable to FLOSS development teams. The knowledge-based view presents coordination, shared mental models,

Table 1. Constructs for study: Embodiments of structures and actions guided by and that reinforce or modify structures.

Structure	Structural embodiment	Actions guided by structure	Actions that create/reinforce/modify structure
Signification	Shared mental models	Task contribution and coordination	Socialization Conversation Recapitulation
Domination	Roles with differential access to resources	Task contribution and coordination	Role definition Role assignment
Legitimation	Norms Formal rules and procedures	Task contribution and coordination	Rule creation and change

communication and decision-making and learning as interdependent issues affecting the effectiveness of distributed teams. Grant suggests that to integrate knowledge, firms need coordination mechanisms including rules, sequencing and routines that economize on communication, knowledge transfer and learning, and team decision making and problem solving for the most complex and unusual tasks. Finally, although there is differentiation between experts in what they know, Grant identifies shared mental models as an important prerequisite for knowledge integration.

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 dynamics of the formation and evolution of distributed teams of FLOSS developers, we will carry out a longitudinal in-depth multiple case study design, as suggested by Barley and Tolbert [1]. The overall research design is shown in Figure 3. As Yin defines it, a case study is “an empirical inquiry that investigates a contemporary phenomenon within its real-life context; when the boundaries between phenomenon and context are not clearly evident; and in which multiple sources of evidence are used” [102]. For some of the cases (cases 1 and 2 in Figure 3), we will combine the longitudinal study with retrospective data analysis.

We will examine the dynamics of changes for each of the aspects of structure identified in Table 1 (shared mental models, roles, rules and norm). Each case study will draw on multiple sources of data, including observation and participant observation, project and developer demographics, project plans and procedures, and interviews. The data will be analyzed using content analytic techniques, cognitive maps, process maps and social network analysis. Throughout the study, we plan to check our design and preliminary results with frequent engagement with the FLOSS community through a project advisory board of developers.

Barley and Tolbert [1] 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).

In the remainder of this section, we will 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. In selecting teams to study, we will consider theoretical and pragmatic aspects.

- ∞ First, we will pick projects that vary in the level of corporate participation, for the reasons discussed above in the conceptual development section.
- ∞ Second, 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. First, Barley and Tolbert [1] note the difficulties of identifying settings that are likely to experience interesting changes. Second, 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

Step two: Charting flows of actions. In this step we extract the interactions of team members within a particular time period to investigate the dynamics by which the teams develop over time. We plan to interview developers for each case at least every six months (see 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 the dynamics of the teams. More specifically we look to uncover the patterns of behavior through which members change shared mental models, roles, and norms and rules. We investigate the dynamics by which teams develop shared mental models by studying how members contribute to and coordinate tasks paying special attention to evidence of recapitulation, socialization, conversation. We study how role are assigned and evolve over time by studying member contribution and looking for evidence of role definition and role changes. Lastly, we 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 [1] 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 dynamics 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 (Table 1), we will collect 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 2 shows the mapping from each construct to data source.

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 rules of members using this source.

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 [96]. For example, Scacchi [7] 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 [32,67]. These data are useful because they are unobtrusive measures of the team's behaviours [105]. Mailing list archives will be examined, as email is a primary tool used to support team communication, learning and socialization. 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. 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. We will examine features request archives and logs from other interaction tools, such as chat sessions. 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. Mailing list archives is the main source of interaction data that illuminates the ‘scripts’ for the analysis of dynamics [1]. Observation data from email logs can potentially provide a rich description of the behaviors (patterns of interaction) of FLOSS teams. This rich description leads to a better understanding of the dynamics of FLOSS development.

Observation. We have found from our initial pilot study (described below under Results from Prior Funding) that developers interact extensively at conferences. Indeed, Nardi and Whittaker [106] 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 [107]. 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 as an opportunity to observe and document the role of face-to-face interaction for FLOSS teams.

Participant observation. We plan to carry out a virtual ethnographic study of developer socialization and interaction relying on participant observation of the teams. One student involved with the project has already virtually joined several development teams (with the permission of the project leaders and the knowledge of other members) and is currently participating in their normal activities while observing and recording these activities (following a protocol approved by the Syracuse University Human Subjects Review Board). In this way, we will study and learn

Table 2. 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

Table 3. 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
Participant observation	Content analysis, process mapping, cognitive mapping

first hand the socialization and coordination practices of these teams. We will track these teams through the various stages of development status, from planning through production/stable stage, observing how new members join the teams and how they contribute to the team output.

Developer interviews. While the data sources listed above will provide an extensive pool of data, they are mostly indirect. Interviews are important to get rich, first-hand data about developers' perceptions and interpretations. We plan to conduct interviews with key informants in the selected projects. Interviews will be conducted in part by e-mail, but we also plan to attend one or two FLOSS conferences each year (e.g., the *O'Reilly Open Source Convention* or *ApacheCon*) to interview FLOSS developers face-to-face. The first round of interviews will be scheduled after the initial data analysis to ensure that we have a sufficient understanding of the process to be able to pose intelligent questions, and on a recurring basis to provide insight into the dynamics of the team, as discussed above. 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' participation [108,109]. 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 described in Section 2 and to address our research questions. Table 3 shows the mapping from data sources to data analysis techniques.

Content analysis. The project will rely heavily on content analysis of the text in interaction archives and interviews to develop insights on the extent and development of shared mental models and socialization (e.g., the way projects are created, introduction of new members, members leaving and community building). Data will be analyzed following the process suggested by Miles and Huberman [110], iterating between data collection, data reduction (coding), data display, and drawing and verifying conclusions. The researchers will develop an initial content analytic framework to discover the patterns of the various variables present in the data. The initial (deductive) framework will be based on indicators from content analytic frameworks previously used to investigate shared mental models [e.g., 111]. In addition we will incorporate work on Asynchronous Learning Networks investigating social, cognitive and structuring processes of virtual teams [112]. 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 [110]. 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 reli-

ability of the coding scheme we will conduct intercoder reliability tests and modify the content analytic scheme until we reach an 85% agreement level [113].

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. Madey, Freeh & Tynan [114] applied this technique to connections between projects, but not within projects. We are particularly interested in using social network information to identify various structural roles in the team 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 [13], the way contributions are distributed among developers and the roles assumed by core developers. The results of such analyses will support us in the identification of the social relations patterns and the way such patterns develop and affect team learning and socialization.

Process maps. The open source software development *processes* will be mapped based on an inductive coding of the steps involved. 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 [115] describe in detail the methods they used to develop and test a process theory of how innovations develop over time. Yamauchi et al. [116] coded messages to understand the development processes of two FLOSS projects. Process traces can be clustered using optimal matching procedures [117] 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 [118].

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.

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 [119-122]. 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 eventual shared mental models acting within teams. 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 [123] 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 3300 hours/year (4400 hours in two years). One of the graduate students will support the principal investigator in sample section, definition of con-

structs and variables, and will have primary responsibility for data collection and analysis, under the oversight of the PI. The second graduate student will be assigned to carry out a virtual ethnographic study of project teams. The principal investigator 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 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 [56] and her competencies in cognitive mapping [124,125] will be particularly valuable for this project.

4. Conclusion

In this proposal, we develop a conceptual framework and a research plan to investigate work practices 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 distributed FLOSS teams. The study has two main strengths. First, we fill a gap in the literature with an in-depth investigation of the dynamics of developing shared mental models, role 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. Second, we use several different techniques to analyze the team dynamics, 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 conceptual, methodological as well as 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" [10]. 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 [39]. Employing qualitative techniques to understand the process of learning will also be a contribution to the organizational learning methodology [126].

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. 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 a digital 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 [36,37,127,128]. 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 [129] 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, which are taught 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 our findings. In order to improve infrastructure for research, we also plan to make our tools and raw data available to other researchers. The project will promote teaching, training, and learning 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 three NSF grants within the past 48 months. The most recent is IIS-0341475, *SGER: Effective work practices for Open Source software development* (\$12,052, 1 September 2003 to 31 August 2004). This small grant has provided support for travel to conferences (e.g., *ApacheCon*) to observe, interview and seek support from developers and to present preliminary results, and for the purchase of data analysis software, supporting the initial results reported in this proposal. This work has resulted in an accepted conference paper [130], with additional papers in preparation [e.g., 131].

Earlier support came from IIS-9732799 (\$69,997, September 1, 1998 to February 29, 2000) and IIS-0000178 (\$269,967, July 1, 2000 to June 30, 2003), both 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 [132-134] and numerous conference presentations [e.g., 135,136].

The core of the PI's research agenda concerns novel organizational forms enabled by new uses of ICT. The present proposal builds on his interest in coordination processes and virtual organizations by studying a novel setting, namely FLOSS development teams.

References

- 1 Barley, S.R. and Tolbert, P.S. (1997) Institutionalization and structuration: Studying the links between action and institution. *Organization Studies* 18 (1), 93–117
- 2 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
- 3 Raymond, E.S. (1998) The cathedral and the bazaar. *First Monday* 3 (3)
- 4 Wayner, P. (2000) *Free For All*, HarperCollins
- 5 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, ACM
- 6 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)*
- 7 Scacchi, W. (2002) Understanding the Requirements for Developing Open Source Software Systems. *IEEE Proceedings Software* 149 (1), 24–39
- 8 Weick, K.E. and Roberts, K. (1993) Collective mind in organizations: Heedful interrelating on flight decks. *Administrative Science Quarterly* 38 (3), 357–381
- 9 Cannon-Bowers, J.A. and Salas, E. (2001) Reflections on Shared Cognition. *Journal of Organizational Behavior* 22, 195–202
- 10 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, Oxford Press
- 11 Huber, G.P. (1991) Organizational learning: The contributing processes and the literatures. *Organization Science* 2 (1), 88–115
- 12 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, MIT Press
- 13 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, ACM, San Francisco
- 14 Nejme, B.A. (1994) Internet: A strategic tool for the software enterprise. *Communications of the ACM* 37 (11), 23–27
- 15 Scacchi, W. (1991) The Software Infrastructure for a Distributed Software Factory. *Software Engineering Journal* 6 (5), 355–369
- 16 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
- 17 van Fenema, P.C. (2002) Coordination and control of globally distributed software projects. In *Erasmus Research Institute of Management*, pp. 572, Erasmus University
- 18 de Souza, P.S. (1993) Asynchronous Organizations for Multi-Algorithm Problems. Department of Electrical and Computer Engineering, Carnegie-Mellon University
- 19 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, MIT Press
- 20 Curtis, B., Walz, D. and Elam, J.J. (1990) Studying the Process of Software Design Teams. In *Proceedings*, pp. 52–53
- 21 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

- 22 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
- 23 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, MIT Press
- 24 Curtis, B., Krasner, H. and Iscoe, N. (1988) A field study of the software design process for large systems. *CACM* 31 (11), 1268–1287
- 25 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
- 26 Humphrey, W.S. (2000) *Introduction to team software process*, Addison-Wesley
- 27 Sawyer, S. and Guinan, P.J. (1998) Software development: Processes and performance. *IBM Systems Journal* 37 (4), 552–568
- 28 Brooks, F.P., Jr. (1975) *The Mythical Man-month: Essays on Software Engineering*, Addison-Wesley
- 29 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
- 30 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
- 31 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
- 32 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
- 33 Butler, B., Sproull, L., Kiesler, S. and Kraut, R. (In press) Community Effort in Online Groups: Who Does the Work and Why? In *Leadership at a Distance* (Weisband, S. and Atwater, L., eds.)
- 34 Grabowski, M. and Roberts, K.H. (1999) Risk mitigation in virtual organizations. *Organization Science* 10 (6), 704–721
- 35 Herbsleb, J.D. and Grinter, R.E. (1999) Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software* (September/October), 63–70
- 36 Jarvenpaa, S.L. and Leidner, D.E. (1999) Communication and trust in global virtual teams. *Organization Science* 10 (6), 791–815
- 37 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
- 38 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, MIT Press
- 39 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
- 40 Orlikowski, W.J. (2002) Knowing in Practice: Enacting a Collective Capability in Distributed Organizing. *Organization Science* 13 (3), 249–273
- 41 Di Bona, C., Ockman, S. and Stone, M., eds (1999) *Open Sources: Voices from the Open Source Revolution*, O'Reilly & Associates
- 42 Kogut, B. and Metiu, A. (2001) Open-source software development and distributed innovation. *Oxford Review of Economic Policy* 17 (2), 248–264
- 43 Lerner, J. and Tirole, J. (2001) The open source movement: Key research questions. *European Economic Review* 45, 819–826
- 44 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

- 45 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
- 46 Bessen, J. (2002) *Open Source Software: Free Provision of Complex Public Goods* Research on Innovation
- 47 Franck, E. and Jungwirth, C. (2002) *Reconciling investors and donators: The governance structure of open source*, Working Paper (No. 8) Lehrstuhl für Unternehmensführung und -politik, Universität Zürich
- 48 Markus, M.L., Manville, B. and Agres, E.C. (2000) What makes a virtual organization work? *Sloan Management Review* 42 (1), 13–26
- 49 Ljungberg, J. (2000) Open Source Movements as a Model for Organizing. *European Journal of Information Systems* 9 (4)
- 50 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
- 51 Bezroukov, N. (1999) Open source software development as a special type of academic research (critique of vulgar raymondism). *First Monday* 4 (10)
- 52 Bezroukov, N. (1999) A second look at the Cathedral and the Bazaar. *First Monday* 4 (12)
- 53 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
- 54 Stewart, K.J. and Gosain, S. (2001) Impacts of ideology, trust, and communication on effectiveness in open source software development teams. In *Twenty-Second International Conference on Information Systems*, pp. 507–512, New Orleans, LA
- 55 Valloppillil, V. (1998) *Halloween I: Open Source Software*, Available from <http://www.opensource.org/halloween/halloween1.html>
- 56 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
- 57 Prasad, G.C. (n.d.) *A hard look at Linux's claimed strengths...*, Available from <http://www.osopinion.com/Opinions/GaneshCPrasad/GaneshCPrasad2-2.html>
- 58 Valloppillil, V. and Cohen, J. (1998) *Halloween II: Linux OS Competitive Analysis*, Available from <http://www.opensource.org/halloween/halloween2.html>
- 59 Hallen, J., Hammarqvist, A., Juhlin, F. and Chrigstrom, A. (1999) Linux in the workplace. *IEEE Software* 16 (1), 52–57
- 60 Leibovitch, E. (1999) The business case for Linux. *IEEE Software* 16 (1), 40–44
- 61 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>
- 62 Moody, G. (2001) *Rebel code—Inside Linux and the open source movement*, Perseus Publishing
- 63 Vixie, P. (1999) Software engineering. In *Open sources: Voices from the open source revolution* (Di Bona, C. et al., eds.), O'Reilly
- 64 Kraut, R.E. and Streeter, L.A. (1995) Coordination in software development. *Communications of the ACM* 38 (3), 69–81
- 65 O'Reilly, T. (1999) Lessons from open source software development. *Communications of the ACM* 42 (4), 33–37
- 66 Shepard, T., Lamb, M. and Kelly, D. (2001) More testing should be taught. *Communication of the*

ACM 44 (6), 103–108

- 67 Lee, G.K. and Cole, R.E. (2000) *The Linux Kernel Development As A Model of Open Source Knowledge Creation*, Unpublished manuscript Haas School of Business, University of California, Berkeley
- 68 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, Harvard Business School Press
- 69 Argyris, C. and Schön, D.A. (1996) *Organizational Learning II: Theory, method and practice*, Addison-Wesley
- 70 Guzzo, R.A. and Dickson, M.W. (1996) Teams in organizations: Recent research on performance effectiveness. *Annual Review of Psychology* 47, 307–338
- 71 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
- 72 Orlikowski, W.J. (1992) The duality of technology: Rethinking the concept of technology in organizations. *Organization Science* 3 (3), 398–427
- 73 DeSanctis, G. and Poole, M.S. (1994) Capturing the complexity in advanced technology use: Adaptive structuration theory. *Organization Science* 5 (2), 121–147
- 74 Walsham, G. (1993) *Interpreting Information Systems in Organizations*, John-Wiley
- 75 Newman, M. and Robey, D. (1992) A social process model of user-analyst relationships. *MIS Quarterly* 16 (2), 249–266
- 76 Giddens, A. (1984) *The Constitution of Society: Outline of the Theory of Structuration*, University of California
- 77 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
- 78 Gregory, D. (1989) Presences and absences: Time-space relations and structuration theory. In *Social theory of modern societies: Anthony Giddens and his critics*, Cambridge University Press
- 79 Cassell, P., ed. (1993) *The Giddens Reader*, Stanford University Press
- 80 Orlikowski, W.J. and Yates, J. (1994) Genre repertoire: The structuring of communicative practices in organizations. *Administrative Sciences Quarterly* 33, 541–574
- 81 Hackman, J.R. (1986) The design of work teams. In *The Handbook of Organizational Behavior* (Lorsch, J.W., ed.), pp. 315–342, Prentice-Hall
- 82 Finholt, T. and Sproull, L.S. (1990) Electronic groups at work. *Organization Science* 1 (1), 41–64
- 83 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
- 84 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, Lawrence Erlbaum Associates
- 85 Dougherty, D. (1992) Interpretive Barriers to Successful Product Innovation in Large Firms. *Organization Science* 3 (2), 179–202
- 86 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
- 87 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, Jossey-Bass
- 88 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
- 89 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
- 90 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
 - 91 Moon, J.Y. and Sproull, L. (2000) Essence of distributed work: The case of Linux kernel. *First Monday* 5 (11)
 - 92 Cox, A. (1998) *Cathedrals, Bazaars and the Town Council*, Available from <http://slashdot.org/features/98/10/13/1423253.shtml>, accessed 22 March 2004
 - 93 Gacek, C., Lawrie, T. and Arief, B. (n.d.) *The many meanings of Open Source*, Unpublished manuscript Centre for Software Reliability, Department of Computing Science, University of Newcastle
 - 94 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>.
 - 95 Hecker, F. (1999) *Mozilla at one: A look back and ahead*, Available from <http://www.mozilla.org/mozilla-at-one.html>
 - 96 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*
 - 97 Swieringa, J. and Wierdsma, A. (1992) *Becoming a Learning Organization*, Addison-Wesley
 - 98 March, J.G., Schulz, M. and Zhou, X. (2000) *The Dynamics of Rules: Change in Written Organizational Codes*, Stanford University Press
 - 99 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
 - 100 Grant, R.M. (1996) Toward a knowledge-based theory of the firm. *Strategic Management Journal* 17 (Winter), 109–122
 - 101 Grant, R.M. (1996) Prospering in dynamically-competitive environments: Organizational capability as knowledge integration. *Organizational Science* 7 (4), 375–387
 - 102 Yin, R.K. (1984) *Case study research: Design and methods*, Sage
 - 103 Krishnamurthy, S. (2002) *Cave or Community? An Empirical Examination of 100 Mature Open Source Projects* University of Washington, Bothell
 - 104 Hare, A.P. (1976) *Handbook of Small Group Research*, Free Press
 - 105 Webb, E. and Weick, K.E. (1979) Unobtrusive measures in organizational theory: A reminder. *Administrative Science Quarterly* 24 (4), 650–659
 - 106 Nardi, B.A. and Whittaker, S. (2002) The place of face-to-face communication in distributed work. In *Distributed Work* (Hinds, P. and Kiesler, S., eds.), pp. 83–110, MIT Press
 - 107 Edwards, K. (2001) Epistemic Communities, Situated Learning and Open Source Software Development. In *Epistemic Cultures and the Practice of Interdisciplinarity Workshop*, NTNU, Trondheim
 - 108 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, MIT Press
 - 109 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, MIT Press
 - 110 Miles, M.B. and Huberman, A.M. (1994) *Qualitative Data Analysis : An Expanded Sourcebook*, Sage Publications
 - 111 Edmondson, A. (1999) Psychological Safety and Learning Behavior in Work Teams. *Administrative Science Quarterly* 44 (2), 350-383
 - 112 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)*, IEEE Press, Big Island, Hawaii
- 113** 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
- 114** 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
- 115** 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
- 116** 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
- 117** Abbott, A. (1990) A primer on sequence methods. *Organization Science* 1 (4), 375–392
- 118** 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.), MIT Press
- 119** Carley, K.M. and Palmquist, M. (1992) Extracting, representing and analyzing mental models. *Social Forces* 70 (3), 601–636
- 120** Carley, K.M. (1997) Extracting team mental models through textual analysis. *Journal of Organizational Behaviour* 18, 533–558
- 121** Langfield-Smith, K. (1992) Exploring the need for a shared cognitive map. *Journal of management studies* 29 (3), 349–368
- 122** 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
- 123** Crowston, K. and Kammerer, E. (1998) Coordination and collective mind in software requirements development. *IBM Systems Journal* 37 (2), 227–245
- 124** Albino, V., Kultz, S. and Scozzi, B. (2003) Actors and cognitive maps on sustainable development in industrial district. In *Uddevalla Symposium*, Uddevalla, Sweden
- 125** 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
- 126** Miner, A.S. and Mezias, S.J. (1996) Ugly Duckling No More: Pasts and Futures of Organizational learning. *Organization Science* 7 (1), 88–99
- 127** Bélanger, F. and Collins, R. (1998) Distributed Work Arrangements: A Research Framework. *The Information Society* 14 (2), 137–152
- 128** Carmel, E. and Agarwal, R. (2001) Tactical approaches for alleviating distance in global software development. *IEEE Software* (March/April), 22–29
- 129** 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
- 130** 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)*
- 131** 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
- 132** 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

- 133** 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
- 134** 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
- 135** 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
- 136** 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



Politecnico di Bari
DIPARTIMENTO DI INGEGNERIA PER L'AMBIENTE E LO
SVILUPPO SOSTENIBILE

dr. Barbara Scozzi

31 March 2004

Kevin Crowston
Syracuse University School of Information Studies
4-206 Centre for Science and Technology
Syracuse, NY 13244-4100
USA

Dear Kevin:

I would like to be involved as a collaborator in your project "Dynamics of Open Source Software Development Teams". My participation would represent a chance to advance my understandings of the knowledge management dynamics adopted in a computer-mediated environment, while working in a very stimulating international context. I believe that my contribution would also be extremely useful for the project development.

I'm an Assistant Professor of Business and Management Engineering at the Polytechnic of Bari (Italy). My research activity deals with the analysis of coordination and knowledge management practices adopted in business organizations and, in particular, on the role that information systems play to support such practices. Open Source Software (OSS) development teams represent a perfect context where to study those aspects. The teams exemplify a new and successful organizational form enabled by the use of Information and Communication Technology. Investigating the knowledge management dynamics adopted by such teams would provide relevant insights on some aspects, such as learning and socialization, as they occur in a distributed work environment and on the way they affect the work performance.

This project would be a good extension to our current research on OSS project's success factors, some results of which have already been published on an academic journal (please refer to the attached biographical sketch). Our joint research activity develops within a collaborative research agreement established between my University (Polytechnic of Bari - Department of Mechanics and Business Engineering) and the School of Information Studies (Syracuse University). The agreement is aimed at promoting the development of joint research projects, fostering the exchange of scientific knowledge and facilitating the mobility of professors and students. My eventual involvement in the project would, thus, be part of a research program already in progress, for which I'm going to present a proposal project to the European Community and the Italian Minister of Instruction, University and Research.

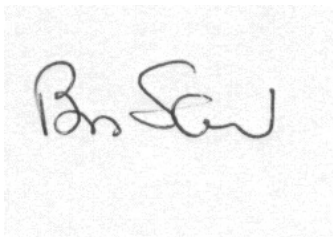
My competencies in the study of knowledge management practices as well as my interest in cognitive mapping could be particularly useful for the project outcomes. In

particular, I would like to contribute to the analysis of the cognitive models adopted within OSS development teams, I could interview (face-to-face and by email) the OSS developers involved in some selected projects so as to develop their cognitive maps. The analysis and comparison among the maps would be aimed at verifying the existence (or the lack of) of a shared perspective about the adopted practices with a specific focus on learning and socialization. Also, I would investigate the relationship among the existence (or the lack of) a collective mind and the work performance.

I would participate to the project both in first person and through the work that some students of the Politecnico of Bari, under my supervision and funded by the Politecnico of Bari, could provide. The students could spend a period at the School of Information Studies to work on the project. That would be an extremely formative experience for them and a very useful support for the project development. This kind of experience has been already experimented by some students of the Politecnico, mr. Giuseppe Sardone (graduate student) and mr. Salvatore Buonocore (undergraduate student), that visited the School of Information Studies (academic year 2002-2003) and worked on issues related to Open Source Software.

Based on the above considerations, I again assert my vivid interest in taking part to the project, as that participation would enhance my research activity and be valuable for the project development.

Barbara Scozzi

A handwritten signature in black ink, appearing to read 'Bar Scozzi', is centered on a light gray rectangular background.