# Project Summary: DHB: Investigating the Dynamics of Free/Libre Open Source Software Development Teams

Kevin Crowston, Syracuse University (with participation from Polytechnic of Bari, Italy)

Increasingly, organizational work is performed by distributed teams of interdependent knowledge workers. Such teams have many benefits, but geographic, organizational and social distance between members makes it difficult 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 distributed teams, especially not self-organizing ones. We propose a multi-disciplinary and inter-disciplinary study (social and computer science) in the context of 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 self-organizing distributed teams develop and work?**

We will study how distributed teams 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 these teams. As a basis for this study, we develop a conceptual framework that uses a structurational perspective to integrate research on team behaviour, communities of practice and shared mental models. A key innovation of this proposal is the integration of three methods to investigate these dynamics: natural language processing and social network analysis of team interactions and source code analysis. The work will be carried out by a multi-disciplinary team including researchers from the fields of information systems and natural language processing, and with participation of an international collaborator at Politechnic of Bari. The research will be guided by an advisory board of FLOSS developers to ensure relevance and to promote diffusion of our findings into practice.

*Expected intellectual merits*

The proposed study will have conceptual, methodological as well as practical contributions. Developing an <u>integrated theoretical framework</u> to understand the dynamics of a distributed team will be a contribution to the study of distributed teams. The project will <u>advance knowledge and understanding</u> of FLOSS development and distributed work more generally by identifying how these teams evolve and how new members are socialized. 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 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. Furthermore, we will use several different techniques to analyze these practices, and thus provide a richer portrait of the dynamics of these development teams.

*Expected broader impacts*

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

# Project Description: DHB: Investigating the Dynamics of Free/Libre Open Source Software Development Teams

Increasingly, organizational work is performed by distributed teams of interdependent knowledge workers. Such teams have many benefits, but the geographic, organizational and social distance between members make it difficult 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. These dynamics are particularly challenging when teams have the autonomy or responsibility to self-organize (e.g., in teams that span organizations). We propose a multi-disciplinary and inter-disciplinary (social and computer science) study 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 self-organizing distributed teams develop and work?**

Our study will be set in the context of teams of Free/Libre Open Source (FLOSS) software developers. 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 FLOSS (e.g., Linux or Apache). 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[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) and even enterprise systems (e.g., eGroupware, Compiere, openCRX).

Key to our interest is the fact that most FLOSS software is developed by dynamic self-organizing distributed teams comprising professionals, users [189-191] and other volunteers working in loosely coupled teams. 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) [160,197]. The teams have a high isolation index [154] in that most team members work on their own and in most cases for different organizations (or no organization at all). For most 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. While these features place FLOSS teams towards the end of the continuum of virtual work arrangements, the emphasis on distributed work makes them useful as a research setting for isolating the implications of this organizational innovation.

The research literature on software development and on distributed work (summarized below in Section 1) emphasizes the difficulties of distributed software development, but the apparent success of FLOSS development presents an intriguing counter-example. What is perhaps most surprising about the FLOSS development process is that it appears to eschew traditional project coordination mechanisms such as formal planning, system-level design, schedules, and defined development processes [8,95]. Furthermore, 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, effective FLOSS development teams somehow profit from the advantages and overcome the challenges of distrib-

1 FLOSS software is usually available 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 unrestrictive license terms ("free as in speech", thus "libre"). We have chosen to use the acronym FLOSS rather than the more common OSS to acknowledge this dual meaning.

uted 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. While in many ways unique, the distributed and self-organizing nature of FLOSS teams represents a mode of work that is becoming increasingly common in many organizations, so results from our study will be broadly applicable.

As well, FLOSS development is an important phenomena deserving of study for itself [67]. FLOSS is an increasingly important commercial phenomenon involving all kinds of software development firms, large, small and startup. Millions of users depend on FLOSS systems such as Linux and the Internet, which is heavily dependent on FLOSS tools. However, as Scacchi [173] 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”. Furthermore, large and longitudinal studies of software development, such as the one conducted by Perry et al. [156], remain rare. 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. The proposed research will be guided by an advisory board of FLOSS developers to ensure relevance of our study to software development and to help promote diffusion of our findings into practice.

To study the dynamics of self-organizing distributed teams, we propose a multi-disciplinary and inter-disciplinary study that integrates analysis of multiple sources of data using multiple research methods. We will use a combination of natural language processing (NLP) and social network analysis (SNA) to analyze large quantities of developer email and chat logs. We will correlate these findings with analysis of the software structure of the code produced by the teams to understand the effects of the team dynamics on the teams’ output. FLOSS teams provide a perfect setting for such a study because large quantities of interaction data and the program source code are readily available for study. The novel mix of research approaches—seldom linked—requires a large and multi-disciplinary research team that does not fit well in existing NSF programs[2]. The proposed research team includes individuals from multiple research fields with expertise in the social dynamics of teams, NLP, qualitative text and social network analysis, and with expertise in FLOSS development. The inter-disciplinary nature of these techniques will provide a rich and more complete picture of the functioning of these teams and will link the behavior of individual members to the outcome of the teams and to their social underpinnings as they evolve over varying time scales. The proposed work will also make a contribution to the underlying fields it draws from. For example, developing techniques to analyze chat transcripts will help progress in NLP field; understanding FLOSS development will contribute to the field of empirical software engineering and information systems.

The remainder of this proposal is organized into five 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, using structuration theory [12] as an organizing framework to integrate theories of shared mental models [27,199], organizational learning [100,132] and team learning [64]. In section 3, we present the study design, with details of the data collection and analysis plans. In this section, we describe how our research will integrate social science, empirical software engineering and natural language processing, and contribute to the improvement of all three. In section 4, we present the project management plan. We conclude in section 5 by sketching the intellectual merits and expected broader impacts of our study and by reviewing results of prior NSF support.

[2] Indeed, the first PI has a proposal under review with a similar aim to this one, but since the restrictions of the traditional NSF program preclude a multi-disciplinary approach, that proposal does not include NLP techniques, but rather relies on manual analysis of just four projects, drawing on a different mix of data. While there are certainly synergies between these projects, neither depends on the other for success. If both proposals were funded, the results would be complementary rather than overlapping.

## 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., 147], 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 [137]. Distributed teams seem particularly attractive for software development because the code can be shared via the same systems used to support team interactions [145,172].

While distributed teams have many potential benefits, distributed workers face many real challenges. Watson-Manheim, Chudoba, & Crowston [196] suggest 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 [187], or that produces unintended information filtering [56] or misunderstandings [7]. These interpretative difficulties, in turn, make it hard for team members to develop shared mental models of the developing project [55,65]. A lack of common knowledge about the status, authority and competencies of team participants can be an obstacle to the development of team norms [10] and conventions [135].

The presence of discontinuities seems likely to be particularly problematic for software developers [187], hence our interest in distributed software development. Numerous studies of the social aspects of software development teams [54,101,171,187,195] conclude that large system development requires knowledge from many domains, which is thinly spread among different developers [54]. As a result, large projects require a high degree of knowledge integration and the coordinated efforts of multiple developers [22]. More effort is required for interaction when participants are distant and unfamiliar with each others' work [149,175]. The additional effort required for distributed work often translates into delays in software release compared to traditional face-to-face teams [96,139]. 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 would be expected to make it hard to achieve an integrated product [95].

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 [24]. Research has shown the importance of formal and informal coordination mechanisms and information sharing [195] 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 [77,94,102,105,108]. Successful distributed teams share knowledge and information and create new practices to meet the task-related and social needs of the members [163]. However, the dynamics of knowledge sharing and socialization for distributed teams are still open topics for research [152].

### *Research on FLOSS development*

The growing 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., 58,107,117]. Second, various explanations have been proposed for why individuals decide to contribute to projects without pay [e.g., 15,69,87,98,136]. These authors have mentioned factors such as increasing the usefulness of the software [88], personal interest [88], ideological commitment, development of skills [130] with potential career impact [88] or enhancement of reputation [136]. Finally, a few authors have investigated the processes of FLOSS development [e.g., 160,178], which is the focus of this proposal.

Raymond's [160] 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 [16,17]. Recent empirical work has begun to illuminate the structure and function of FLOSS development teams.

The other major stream of research examines factors for the success of FLOSS in general terms (though there have been few systematic comparison across multiple projects, e.g., [179]). 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 [48,85,116,157,158,184,185]. In turn, the speed of development and the quality of the software 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 [142,188], 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 [109]. (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. Asklund & Bendix [8] note the resulting importance of well-written and easy-to-read code.

One of the PIs on this proposal, Kevin Crowston, has been active in FLOSS research, supported by NSF grant IIS 04–14468 ($327,026, 2004–2006), for *Effective work practices for Open Source Software development*, which continued SGER IIS 03–41475, ($12,052, 2003–2004). The initial results of this funding include an analysis of FLOSS teams as virtual organizations [48], theoretical models of FLOSS team effectiveness [36,38] and leadership [39] and a study of possible success measures for FLOSS [35,37]. Empirically, Crowston and his team have analyzed the problems in using data from SourceForge [99], carried out social network analyses to understand the centralization and the hierarchy of project teams [41,42] and described the role of face-to-face meetings in FLOSS teams [43]. These earlier grants are aimed at identifying work practices that characterize effective FLOSS teams. In the research proposed here, we propose to carry out a larger-scale study of the dynamics of FLOSS teams. We have chosen this new focus because studies of FLOSS teams (including our own) and of distributed teams more generally point to the need to understand dynamics of technology-supported self-organizing distributed teams.

## 2. Conceptual development

In this section we develop the conceptual 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. Other researchers have described them as communities of practice, which is a useful lens for studying how knowledge and practices are shared (as we discuss below). However, for the purpose of our study, we view the projects as entities that have a goal of developing a product, a user base to satisfy and a shared social identity. Project members are interdependent in terms of tasks and roles and core members know and acknowledge each other's contributions. These aspects of FLOSS projects suggest analyzing them as work teams. Guzzo and Dickson [84] 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 coworkers)". A team differs from a community of practice because members have a shared output whereas in communities of practice, (e.g., the copier repairmen studied by Orr [153]), members share common practices, but are individually responsible for their own tasks.

*A structurational perspective on team dynamics*

To conceptualize the dynamics of these teams and the process of changes within them, we adopt a structurational perspective. Numerous authors have used a structurational perspective to support empirical analyses of group changes [11,57,146,150,193], 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 [150] and Barley and Tolbert [12]. Structuration theory [74] 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 (i.e., systems of signification, domination and legitimation that influence individual action) 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, meaning that the structural properties of a social system are both the means and the ends of the practices that constitute the social system. As Sarason [168] 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 [81]. Barley and Tolbert [12] note that structuration is "a continuous process whose operations can be observed only through time" (p. 100). Figure 1, adapted from [12], 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 Cassell [32] 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 [81].

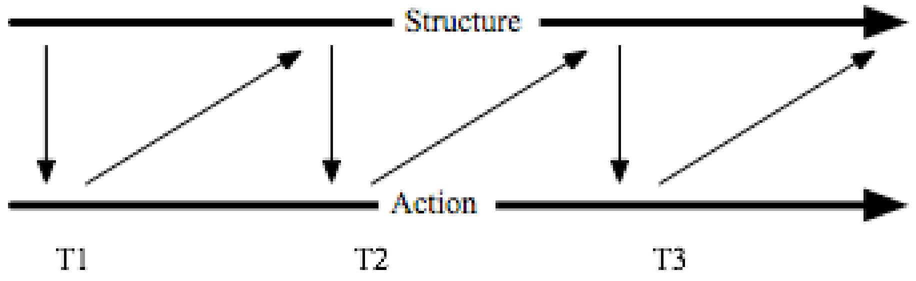

**Figure 1.** A sequential model of the relation between structure and action [from 12].

*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 consider three kinds of rules and resources that are "encoded in actors' stocks of practical knowledge" [12] and "instantiated in recurrent social practice" [151] in the form of interpretive schemes, resources, and norms [12,177]. 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. [26], "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 [93], which describes how individuals know in particular where to find information. That theory was originally developed to explain the behaviours of intimate couples, but recently extended to groups [141] and distributed teams [82,133]. However, research indicates that transactive memory converges to shared mental models as "individuals develop a shared conceptualization of 'who knows what.'" [21]. Yoo & Kanawattanachai [202] similarly argues that transactive memory can develop to collective mind [199]. In our work, we therefore build on the broader concept.

Research suggests that shared mental models help improve performance in face-to-face [162] and distributed teams [180]. Shared mental models can enable teams to coordinate their activities without the need for explicit communications [44,66]. Without shared mental models, individuals from different teams or backgrounds may interpret tasks differently based on their backgrounds, making collaboration and communication difficult [59]. 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 [118,167]. Curtis et al. [55], 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 FLOSS teams in different phases of the software development process [63,73]. 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 [194] 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 [23], 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 through a process of <u>socialization</u>, e.g., by following a "joining script" [192]. Members need to be encouraged and educated to interact with one another to develop a strong sense of "how we do things around here" [93]. Barley and Tolbert [12] similarly note that socialization frequently "involves an individual internalizing rules and interpretations of behaviour appropriate for particular settings" (p. 100). Second, <u>conversation</u> 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 [118]. 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 [23] 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 [141]. A few empirical studies in this area [e.g., 118,162] have investigated the relationship between team or organizational factors and the presence of shared mental models. This study will investigate the process through which members of distributed teams develop shared mental models. This will be accomplished through the analysis of interaction data for evidence of socialization, conversations and recapitulation of 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 [9] (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 [174] or onion-like structure [34,70,143,164], 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 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. Most developers know and acknowledge each other's contributions. The core is usually small (e.g., 9 [103], 11 [106] or 15 [139] developers) and there is a high level of interaction among core members, which would be difficult to maintain if the core 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 traditional teams: in a study of 182 work teams, Cummings and Cross [53] found that core-periphery and hierarchical team structures were negatively associated with performance. On the other hand, Halloran & Scherlis [86] 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). Some might argue that this last group should not be considered as

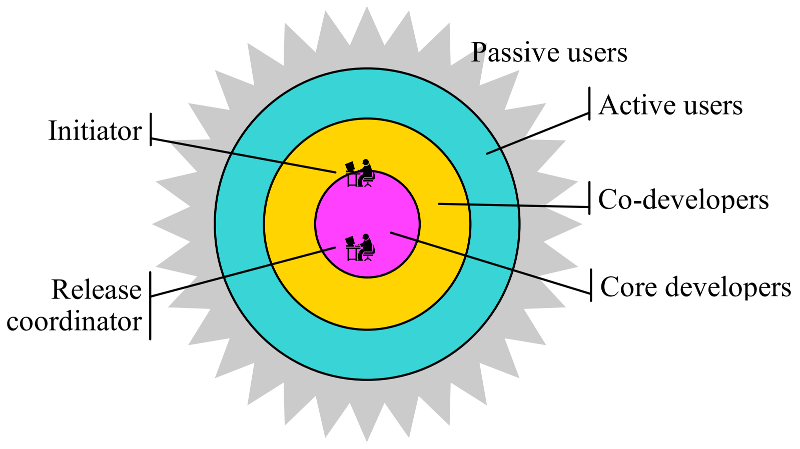

**Figure 2.** Hypothesized FLOSS development team structure.

part of the team, though as we will discuss, they play an important part in the FLOSS development process. Still further from the core are the passive users, who use the project's outputs but are not otherwise part of the project. 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.

There is some evidence that clear definition of these roles is important for project effectiveness. Sutanto, Kankanhalli & Tan [180] found that role ambiguity in distributed teams leads to duplicate work (though this is often not viewed as a problem in FLOSS teams). Sagers [166] argues that restricting access to the core improves coordination and success of project. Halloran & Scherlis [86] 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 [155]. 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 [176]. The FLOSS process enables hundreds of people to work on these parts of the process [115], what Rossi [164] describe 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 [106] 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 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 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, particularly in a distributed team [68,91]. This barrier is particularly troubling because of the reliance of FLOSS projects on volunteer submission of new code and on "fresh blood" [52]. 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 [194], describes one aspect of team functions as the creation of implicit norms and explicit rules [181]. Rossi [164] notes that rules allow developers to form stable expectations of others' actions, thus promoting coordination. The importance of such rules have been documented in conventional software and FLOSS development teams [e.g., 169,179]. For example, Jørgensen [103] describes a set of implicit and explicit rules for software development in the FreeBSD project (e.g., "Don't break the build"), while Raymond [161] notes implicit rules regarding project forking at the community level. Gallivan [72] analyzes descriptions of the FLOSS process and suggests that teams rely on a variety of social control mechanisms rather than on trust. To conceptualize this aspect of teams, we also draw on Swieringa and Wierdsma's [181] 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.

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 results of integrating the knowledge of experts into the team's structure reflecting behavioral changes within a team over time, what March et al. [134] and Hayes and Allinson [90] refer

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

| | Constructs for study | |
|---|---|---|
| **Structure** | **Structural embodiment** | **Actions that create/ reinforce/modify structure** |
| Signification | Shared mental models | Socialization<br>Conversation<br>Recapitulation |
| Domination | Roles with differential access to resources | Role definition<br>Role assignment |
| Legitimation | Norms<br>Formal rules and procedures | Rule creation and change |

to as learning on the group level. However, the practices by which these rules can be developed in distributed settings is an open issue.

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 create, reinforce or modify the structures. The resulting model is largely consistent with Grant's knowledge-based view of the firm [79], which analyzes a firm as a structure for integrating specialist knowledge into the firm's activities and products [78]. 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, 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 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. We first discuss the goals and general design of the study. We then present the details of how data will be elicited and analyzed.

To study the dynamics of the formation and evolution of distributed teams of FLOSS developers, we develop an innovative multi-disciplinary approach to the study of human and social dynamics. For each project, we will draw on multiple sources of data: developer interactions, project and developer demographics, project plans and procedures and the source code. The data will be analyzed using social network analysis (SNA), and cognitive and process maps based on content analysis using Natural Language Processing (NLP) techniques to reveal the dynamics of changes for the aspects of structure identified in Table 1 (shared mental models, roles, rules and norms).

We envision our entire research project as having three overlapping phases. Each phase will last roughly a year, though the transition between these phases will be gradual rather a sharp boundary. The overall design is shown in Figure 3. In the first phase (roughly year 1), we will examine project transcripts manually for evidence of the aspects of structure identified in Table 1 to determine what kinds of evidence will be good candidates for identification using NLP techniques. In parallel, we will specialize our proven NLP techniques to deal with novel kinds of text

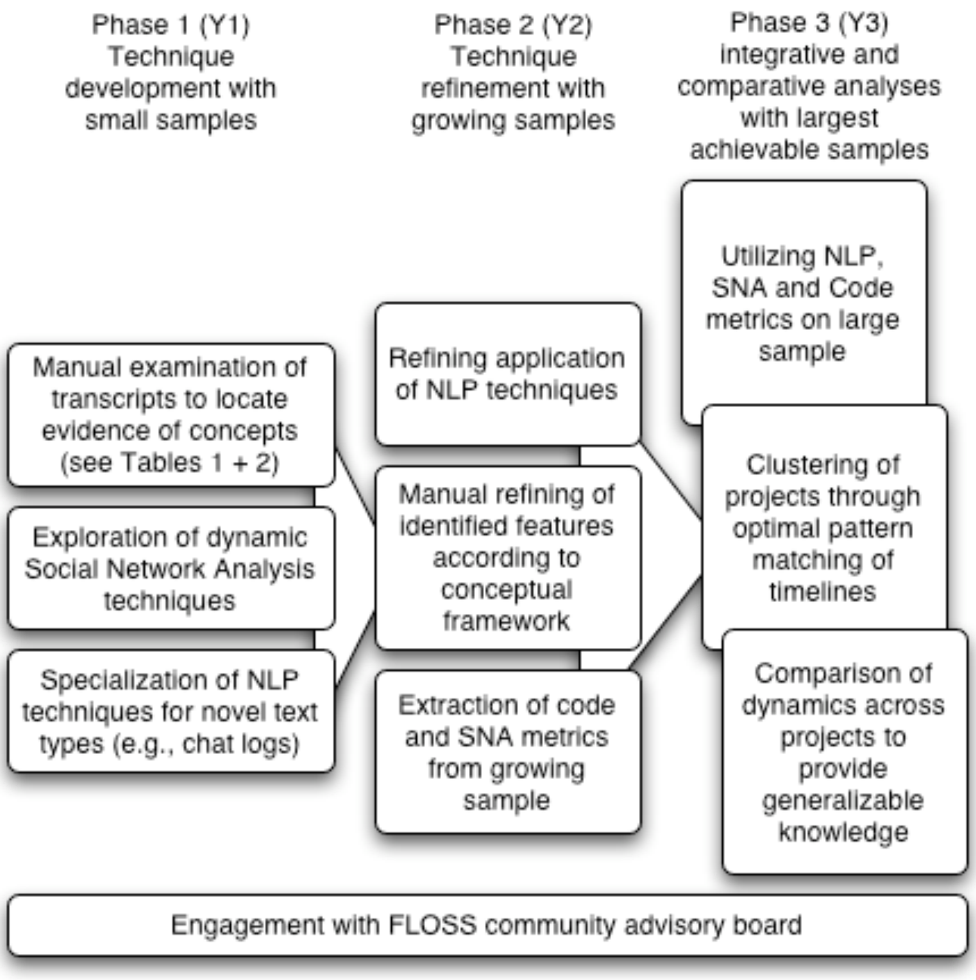
Phase 1 (Y1) Technique development with small samples
Phase 2 (Y2) Technique refinement with growing samples
Phase 3 (Y3) Integrative and comparative analyses with largest achievable samples
Manual examination of transcripts to locate evidence of concepts (see Tables 1 + 2)
Exploration of dynamic Social Network Analysis techniques
Specialization of NLP techniques for novel text types (e.g., chat logs)
Refining application of NLP techniques
Manual refining of identified features according to conceptual framework
Extraction of code and SNA metrics from growing sample
Utilizing NLP, SNA and Code metrics on large sample
Clustering of projects through optimal pattern matching of timelines
Comparison of dynamics across projects to provide generalizable knowledge
Engagement with FLOSS community advisory board

**Figure 3.** Phases of research plan.

such as chat transcripts and identify appropriate dynamic SNA and code analysis techniques. In the second phase (roughly year 2), we will use the NLP techniques to extract larger numbers of the identified research-relevant features and will begin to correlate these with each other and with the results of SNA and source code analysis. In the final phase (roughly year 3), we will analyze large numbers of projects to develop generalizable findings. Throughout the study, we plan to check triangulate our design and preliminary results with frequent engagement with the FLOSS community through a project advisory board of developers.

In each phase, we will follow the four-step process suggested by Barley and Tolbert for applying a structurational perspective to analyzing organizational change [12]:

"(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 each phase by identifying promising FLOSS projects for study. During the first phase, we will our study a small number of teams, increasing the sample in subsequent phases. In the final phase, the size of the sample will be limited only by the available data and processing power (computer and human). In order to ensure that we are studying teams large enough to have coordination problems (as opposed to single person development efforts [110]), we will choose only projects with more than seven core developers [89]. We will include both mature and newly forming teams to be able to assess the initial development stages, though a significant advantage of studying FLOSS teams is the ability to collect data from through out the projects' lifespan. We will also take into consideration some pragmatic considerations, such as selecting only projects where we have access to the data we need (e.g., message logs).

*Step two: Charting flows of actions.* In this step, we analyze the actions of team members within a particular time period. We will extract team interactions from email logs and other interactions and identify team outputs by examining the code created. The analysis will also reveal the structural patterns that prevail at different points in time. The details of data elicitation and analysis are discussed in the following sections.

*Step three: Identifying patterns of changes.* Once we extract segments of interactions and outputs discussed in step two, we will analyze them to reveal the dynamics of the teams. More specifically we will uncover the patterns of behavior through which members change shared mental models, roles and norms and rules, and the implications of these changes for team actions and outputs. We will investigate the dynamics by which teams develop shared mental models by studying how members contribute to and coordinate the tasks, paying special attention to evidence of socialization, conversation and recapitulation. We will study how roles are assigned and how they 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, paying special attention to evidence of rule creation and modification. For each of these types of structure, we will identify how they affect task contribution and coordination.

*Step four: Linking changes in structures to other changes.* In Step 4, Barley and Tolbert [12] suggest linking changes in the structures to other changes of interest in the sites being studied. To accomplish this step, we will triangulate evidence about the teams gathered from multiples sources of evidence about the teams. For example, the implications of changes in structure will be assessed by link them to changes in the program source code and other team outputs or to team membership.

*Data collection*

To explore the concepts identified in the conceptual development section of this proposal (Table 1), we will collect and analyze a range of data: project and developer demographics, interaction logs, project plans and procedures, and source code. In the remainder of this section, we will briefly review each source. Table 2 shows the mapping from each construct to data sources and analysis techniques.

*Developer demographics.* We will collect basic descriptive data about developers, such as their areas of expertise, formal role, years with the project or the other projects in which the developer participates. Often these data are self-reported by the developers on project or personal home pages. We will track changes in the formal roles of members using this source. By examining PGP key signatures, we can identify meetings between developers [148], 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 that is used to keep track of the agenda and working plans [52]. For example, Scacchi [173] 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 requests. 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 teams' interactions for FLOSS development work [96,115]. These data are useful because they are unobtrusive measures of the team's behaviours [198]. Mailing list archives will be a primary source of interaction data that illuminates the 'scripts' for the analysis of dynamics [12], as email is one of the primary tools used to support team communication, learning and socialization [114]. Such archives contain a huge amount of data (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. In a similar analysis of student messages, Dutoit & Bruegge [61] 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. 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.

*Source code.* A major advantage of studying open source software is that we have access to the team outputs in the form of the program source code. As Harrison puts it, "For a change, we [software engineering researchers] can now focus on the analysis rather than the data collection". Most projects use a source code control system such as CVS, which stores intermediate versions of the source and the changes made. From these logs, we will be able to extract data on the date and name of the contributors, the kinds of contributions they make and the change to the source code in order to understand the software structure and the role of individual developers [71,80,140]. Raw software code poses numerous challenges to interpretation [182]. For example, not all projects assign authorship in the CVS tree. We hope to leverage our analysis with work being carried out by other researchers in order to deal with these challenges [e.g., 106].

*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 to raise the level of conceptualization to fit our theoretical perspective. To do so, we are planning a multi-stage analysis process, as

**Table 2.** Constructs, sources of data, and analysis.

| **Concept** | **Constructs** | **Data sources and analysis** |
|---|---|---|
| Action | Task coordination and contribution | Process mapping, social network analysis, code analysis |
| Structures of signification | Shared mental models | NLP-based content analysis of interaction logs |
| | Socialization<br>Conversation<br>Recapitulation | NLP-based content analysis of interaction logs |
| Structures of domination | Roles with differential access to resources | Process mapping, social network analysis, code analysis<br>NLP-based content analysis of interaction logs |
| | Role definition<br>Role changes | Process mapping, social network analysis, code analysis |
| Structures of legitimation | Norms<br>Formal rules and procedures | NLP-based content analysis of interaction logs<br>Project plans and procedures |
| | Rule creation and change | NLP-based content analysis of interaction logs |

shown in Figure 4. These stages will be carried out in some form for each project and in each phase of the research. In the first stage, we will use content analysis, SNA and code metrics to extract relevant phenomenon from the raw data. In the second stage, we will use these results to identify the constructs described in Section 2. The analysis will paint a picture of each project in terms of the contributions towards effective software development as well as towards development of structures of shared mental models, roles, rules and norms. The final stage is to develop process maps that document the flows of action (Step 2) and the patterns of change (Step 3) that address our research questions. These results will show the practices in each project that build and evolve these structures as team members learn to work together and to innovate more effectively. In the remainder of this section, we will describe the analysis approaches to be used in each stage.

Analysis stage 1

The first stage includes three analysis techniques to reduce the large amount of raw data to more specific codes and measures: content analysis, social network analysis and source code analysis.

*Content analysis.* Content analysis of computer-mediated communication (CMC) has been an active area of research [13,97]. This project will rely heavily on content analysis of the text from these interaction archives 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, departure of members and community building).

In the first phase of the research project, data will be content analyzed following the process suggested by Miles and Huberman [138], iterating between data collection, data reduction (coding), data display, and drawing and verifying conclusions. The researchers will develop an initial content analytic framework to uncover the patterns of the concepts 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., 62]. In addition, we will incorporate work on

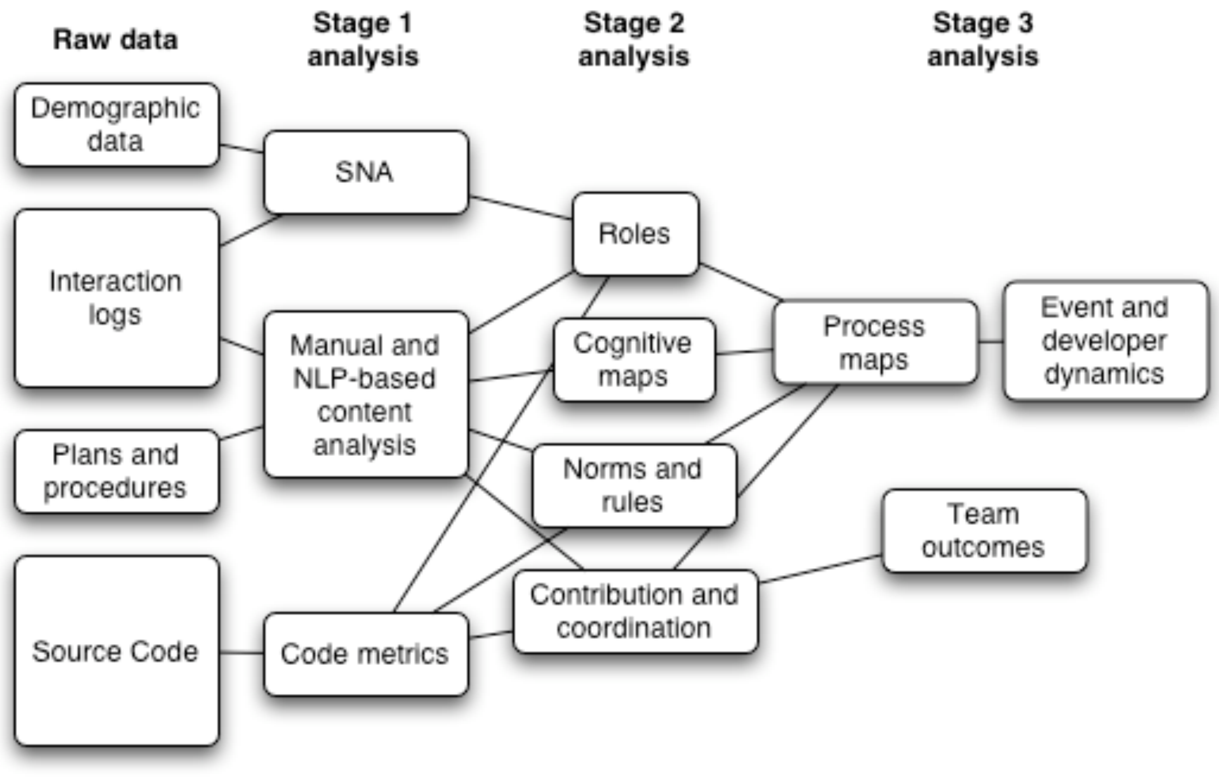

**Figure 4.** Data analysis, from raw data to team dynamics and outcomes.

Asynchronous Learning Networks investigating social, cognitive and structuring processes of virtual teams [92]. However, these manual techniques require a lot of work on the part of the researcher, which limits the amount of data that can be analyzed.

In subsequent phases, we will utilize Natural Language Processing (NLP) technology to assist in identifying important semantic patterns that can then be translated into emerging codes. Turner et al. [183] similarly used some simple NLP approaches to analyze bug reports, though our proposed work goes well beyond this initial effort. Because the use of NLP techniques is one of the major innovations of this proposal and is the foundation of further analysis, we will explain its application in more detail. The NLP-based system developed at the Center for Natural Language Processing (CNLP) at Syracuse University analyzes naturally occurring texts (documents, transcribed interviews, email, chat, etc.) for the explicit and implicit meanings which are conveyed (and which a human would recognize). The resulting NLP annotations will be used as initial codes representing the events, roles, intentions, goals, expectations, etc. reported and/or hinted at in the text (e.g. names, popular abbreviations, special terms, time expressions and other phrases with particular semantic values relevant to the research agenda).

Application of NLP-based text processing for CMC transcripts (e.g., chat room conversations or emails) has been a challenge given the nature of these interactions. These texts are known for their use of specialized language patterns, as well as informal grammar and spelling rules [159]. To effectively meet the challenge of understanding these stylistically diverse and grammatically inconsistent texts, our NLP technology will leverage theoretical and empirical advances in research on Sublanguage Analysis and Discourse Structure. A sublanguage is defined as the particular language usage patterns, which develop within the written or spoken communications of a community that uses this sublanguage to accomplish some common goal or to discuss topics of common interest. Early research in Sublanguage Theory [83,128,129,165] has shown that there are linguistic differences amongst various types of discourse (e.g. news reports, email, manuals, requests, arguments, interviews) and that discourses of a particular type that are used for a common purpose within a group of individuals exhibit characteristic linguistic (lexical, syntactic, semantic, discourse, and pragmatic) features. Humans use these characteristics to extract meaning, and these human processes can be simulated by a full-fledged NLP system in order to extract levels of meaning beyond the simple surface facts.

The fact that a sublanguage deals with a restricted domain and is used for a specific purpose results in useful restrictions on the range of linguistic data that needs to be accounted for by the system. At the lexical level, the sublanguage excludes large parts of the total vocabulary of a language; for those words in the sublanguage vocabulary, the number of senses actually used for each word is limited. At the syntactic level, a sublanguage is characterized by predictable surface structures, utilizes a limited range of verbs, and makes extensive use of domain-specific nominal compounds, which reflect the specialized nature of the sub-field. The discourse level of a sublanguage deals holistically with units of language larger than a sentence, relying on the predictable structure of communications in this sublanguage. The discourse level model of a particular communication type consists of semantic categories (reflecting the purpose of communication) and the relations among those categories. The NLP system's recognition of these semantic categories handles the great surface variety in terms of lexical and syntactic choices in how entities (e.g. people, organizations), events (e.g. updates, requests), and relations amongst them (e.g. who requests an action by whom) are realized in text. As a result, the sublanguage analysis is able to abstract up from these individual instances to the underlying concepts that indicate patterns and reveal trends. Communication types that have been analyzed and for which sublanguage grammars have been developed include abstracts, news articles, arguments, instructions, manuals, dialogue, instructions, email, and queries [129]. The sublanguage analysis framework will be applied to automatically identify the important linguistic patterns in the text-based electronic communications among the FLOSS developers and to annotate them with initial content categories, which will then be refined by the project team to reflect the conceptual framework emerging from data.

*Social network analysis (SNA).* Social network analysis 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 its impact on team outcomes. Madey, Freeh & Tynan [131] applied SNA to connections between projects, but not within projects. Ducheneaut [60] examined interaction patterns, but focused on visualization of the networks. Our work using the SNA approach to interactions in bug fixing logs has revealed that projects display a surprising range of centralizations [41] and most projects were quite hierarchical [42], similar to the results of Ahuja & Carley [2]. However, these analyses have just scratched the surface.

By documenting the social network of a project, we will assess each individual's centrality to the project and the project's level of hierarchy, which seems to mediate the effect of role and status on individual performance within virtual teams [3]. We will also examine 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. Dynamic SNA, the study of the development of networks over time rather than at single static snapshots, is a developing area of research, so our work in this area has the potential to make a contribution to the field in the form of new methodological tools.

*Software source code analysis*. In analyzing the teams' output, we will focus on the program software source code, though outputs such as documentation are also of interest and available for analysis. Analysis of a team's output is important both for assessing the team's performance and for studying the connection between the team's internal evolution and what it actually does. The available documentation in a project as well as the change in the documentation would help us understand the group dynamics and the evolution of the norms in the FLOSS teams. Document analysis is especially useful when analyzed together with the communication logs such as email exchanges or chat logs. Generally speaking, the cost and quality of a software program is linked to the software code's complexity. Software should be only complex enough to solve the problem at hand or to perform the task it is meant to. Additional complexity results in higher costs, in terms of effort, resources and time, and as well as lower quality in the form of, in defects, unpredictability and difficulty in maintenance. Some common metrics for the complexity of a code base include measurements of size (in lines of code or 'function points'), and the coupling and cohesion among the software modules. There are many sets of such measurements in the literature, adapted for the structural type of language, including the Cocomo metrics [18,19] and the "CK" suite of metrics [33].

While the majority of the work in this area involves measuring a static code-base and making and testing predictions regarding its development, there is also a growing body of literature concerned with the evolution (i.e., patterns of change over time) of software projects. Beginning with the work of Belady and Lehman [14], this work takes as its unit of analysis a change in the code made by a developer, paying particular attention to the 'change logs' and 'check-in comments' made by the developers at the time. In our analysis, we will be able to assess these changes and link them back to the discussions in the mailing lists and other developer activities. This work, therefore, crosses the boundary of the code and measures the work practices of individuals and their effects over time, again expressed in terms of complexity, size, faults, and ultimately in software performance [104]. Assessing the evolutions and patterns of change in the code base will provide an additional dynamic element to juxtapose against changes in the organization of the team and the teams' structures of signification, domination and legitimation. The shape of the code base is both structured by the actions of the team and, in turn over time, comes to structure those actions.

Analysis stage 2

In the second stage of the analysis, we will build on the results of the first stage to provide evidence for the concepts in our model: developer activities that contribute to output and team coordination and that build team structures in the form of shared mental models, roles and norms and rules.

*Individual contributions and coordination.* The open source software development processes will be mapped based on a content-analysis coding of the steps involved [200]. 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 in the source code. Yamauchi et al. [200] similarly coded messages to understand the development processes of two FLOSS projects, while Bonneaud, Ripoche & Sansonnet [20] analyzed bug report messages for the Mozilla project to understand the bug fixing practices. 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).

*Shared mental models.* To document shared mental models, we will develop cognitive maps from content-analyzed interaction data. Development of these maps will enable us 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 [29,30,113,144]. We are particularly interested in how these maps evolve over time. Metrics (e.g., number of heads, tails, domain and team 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, Kevin Crowston, has some experience in studying mental models [44] but for this analysis in particular we will work with a collaborator experienced in cognitive mapping, Professor Barbara Scozzi of Polytechnic of Bari, as discussed below.

*Roles.* We plan to document roles using several approaches. First, we will look for descriptions of formal roles and role assignments. Second, we will identify which individuals perform which activities to identify different informal roles. For example, the NLP-based sublanguage analysis will provide subtler indications that implicitly suggest informal roles, as it is based not only on who communicates to whom, but the semantic and affective content of their communications. Finally, we will use social network information to identify various structural roles in the team (e.g., via blockmodels of interactions). In all cases, we will be interested in how individuals fill these roles over time. This analysis of informal and structural roles should provide a useful counterpoint to descriptions of formal roles.

*Norms and rules.* The final concepts in our model are norms and rules. We expect to be able to identify formal rules from the coding of developer interactions and project documentation. Norms may be identified by looking for expressions of implicit standards or moral censure when norms are violated. Identification of coordination mechanisms will also help reveal explicit or implicit rules used by the teams.

### Analysis stage 3

In the third stage of analysis, the results of the analyses discussed above will be integrated to provide the fullest picture of the dynamics of the FLOSS teams. This step corresponds to steps 3 and 4 in Barley and Tolbert's [12] framework. The initial method for integrating the results will be to develop a timeline for each project that show how the activities revealed by each analysis and our inference about the state of the different kinds of structures are related in time. Van de Ven and Poole [186] describe in detail the methods they used to develop and test a process theory of how innovations develop over time.

We will then dissect the timelines to document the history with the project of individual team members and the history of various key events, such as bug fixes, new features or changes in structures. For example, the timeline might show an individual first taking part in team discussions at one point in time, continuing to interact with other members and later contributing code or other products to the project. The history might also include prior discussion that the individual might have been following as a lurker[3], based for example on their initial account creation date. The analysis will provide indications of that individual's sharing of mental models or contributing to their development, the different roles filled over time and evidence of knowledge of or contribution to norm and rule development.

These dissected descriptions can then be clustered and aggregated, e.g., to show typical patterns of participation in a project or different processes for bug fixing or feature development. Process traces can be clustered using optimal matching procedures [1] to develop clusters of processes. Differences in the quality or quantity of contributions can be correlated back to differences in the dynamics of the teams, e.g., in the level of contributions or the modes of coordination or socialization.

The final step in the analysis is to compare these patterns across projects, e.g., to understand why some projects attract and retain more developer participation or are quicker at fixing bugs or developing high quality software. We can then generalize these results to provide findings at conceptual level that applies to other kinds of teams, e.g., effective modes of socialization or of task assignment using volunteers. Another question we intend to consider 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 [8].

## 4. Management plan

Based on preliminary assessment of the effort required, we are requesting funding for three graduate students, summer support for 3 PIs and 10% support during the year for a research scientist, Nancy McCracken. Crowston and Heckman have PhDs in the field of Management and publish mostly in the Information Systems area (Crowston also works in Organizational Communications). Liddy has a PhD in Information Transfer and McCracken in Computer Science, and both work in the field of Natural Language Processing.

All three PIs, Kevin Crowston, Robert Heckman and Elizabeth Liddy, will work during the summer on project management and research design (0.5 months for Crowston and Heckman and 0.3 months for Liddy) and devote 10% of effort during the academic year to project management and oversight (1/2 day per week, supported by Syracuse University). Each PI will be responsible for designing specific aspects of the project and overseeing work on those aspects. Specifically, Crowston will oversee the SNA and code analysis, Heckman will oversee the coding of developer transcripts and Liddy and McCracken will oversee specialization and application of the NLP techniques. As the project continues, these responsibilities will overlap more as the data are integrated. All three PIs will share in project selection, overall project design and report writing. The first PI, Crowston, will be responsible for general project oversight and reporting to NSF.

A PhD student will support each PI. The graduate students will devote 50% effort during the academic year and 100% effort during the summers, for a total of 3300 hours/year (9900 hours in three 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, under the oversight of the PIs. Each student will have initial responsibility for one aspect of the analysis, as discussed above, but in later phases, as the results from the various sources are merged, we anticipate shifting the assignment of responsibilities.

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 collabora-

[3] A 'lurker' is a subscriber to a mailing list or discussion forum who reads but does not (yet) speak.

tor, 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 first PI on a study of FLOSS project success factors [48] and her competencies in cognitive mapping [4,28] will be particularly valuable for this project.

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. To provide a broad perspective, we wanted the board to include developers from a variety of projects, parts of the world and with different roles in projects. Current advisory board members include Paul Everitt, co-founder of Zope Corporation and executive director of the Plone Foundation, Dirk-Willem van Gulik, president of the Apache Software Foundation, as well as Jeffrey Forman, a developer with the Gentoo Linux project.

In order to build an interdisciplinary community of researchers to meet the challenges of this multi-disciplinary research project, we will employ two main project management techniques. First, we will have regular meetings of the project to share findings and to plan the work. Initially, these will be every other week, but the frequency of meetings will be adjusted depending on our experience and the pace of the work being carried out at the time. These formal meetings of all project participants will augment the regular interaction of the PIs with the students working on the data collection and analysis and expected frequent interactions of the students as they integrate data from the same projects. Second, an initial project activity will be the development of a more detailed timeline (based on the project plan presented above) against which progress will be measured. The budget includes support for PIs and PhD students during summer and academic year to support these activities.

## 5. 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 (What are the dynamics through which self-organizing distributed teams develop and work?), 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 self-organizing distributed teams by identifying the dynamics of distributed FLOSS teams. The proposed study has two main strengths. First, we will fill a gap in the literature with an in-depth investigation of the dynamics 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. Second, we will use several different techniques to analyze the team dynamics, providing different perspectives of analysis and thus a richer portrait of the dynamics of the development teams. Moreover, some of data analysis techniques, particularly natural language processing, have not yet been used with FLOSS teams, and as a result this project will contribute not only to the available methodologies for understanding distributed teams, but also serve to further extend the range of capabilities of sublanguage analysis and natural language processing.

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. Developing a theoretical framework consolidating a number of theories to understand the dynamics within a distributed team is an important contribution to the study of distributed teams.

*Expected broader impacts*

The project has numerous broader impacts. The project will benefit society by identifying the dynamics of teams 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 these teams can serve as guidelines (e.g., for team governance, task coordination, communication practices or mentoring) 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 [25,31,102,108]. 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. Members of our research team have presented our earlier work at osdc.com.au (an Australian FLOSS developers conference) and organized a Bird of a Feather session at the ApacheCon conference. Our results could also potentially be incorporated into the curricula of the professional master's degrees of the Syracuse University School of Information Studies. The results could as well as improve the pedagogy of our courses, 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 improve infrastructure for research, we plan to make our tools and data available to other researchers. Efforts to share data collection are already in place based on the current project, in the form of the OSSMole (http://ossmole.sourceforge.net/), a repository for FLOSS data. As a result of the current proposal, we anticipate being able to share analyzed transcripts as well. The project will promote teaching, training, and learning by students in the research project. These students will have the opportunity to develop skills in data collection and analysis.

As well, the project has an important international component with the participation of Dr. Scozzi of Politecnico di Bari, Italy, and her students. Syracuse University has hosted several visitors from the Politecnico di Bari in the past, and with the support of this grant, we plan to have our students spend time working with Dr. Scozzi. Such international exchanges and collaborations are a tremendous vehicle for expanding the perspectives, knowledge and skills of both teams of scientists. They offer a globalization of research and career opportunities, which contributes to the professional and personal development of the students. These exchanges equip students to understand and integrate scientific, technical, social, and ethical issues to confront the challenging problems of the future.

*Results from prior NSF funding*

The PI for this grant, 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, 2004–2006, and SGER IIS 03–41475, $12,052, 2003–2004) were discussed above 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 three journal papers[41,42,48], multiple conference papers [35,38] and workshop presentations [36,37,40,49,99], with additional papers under review [39,43]. The current grant is sought to continue this research, by applying natural language processing techniques to investigate on a large-scale the dynamics of these teams.

The most recent grant is IIS 04–14482 ($302,685, 2005–2006), for "How can document-genre metadata improve information-access for large digital collections?" (with Barbara Kwasnik). This project started less than two months 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., 45] and conference papers [e.g., 111]. The grant partially supported work on a conference mini-track and journal special issue [112]. Earlier support came from IIS–0000178 ($269,967, 2000–2003), 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 [47,51,170] and numerous conference presentations [e.g., 46,50]. The PIs are currently working with the National Association of Realtors to extend and disseminate these results and are planning a follow-on study.

The Co-PI of this proposal, Elizabeth D. Liddy, has received NSF funding for five projects in the past five years. One is briefly described first, while the remaining four form a cohesive research program, which is described in more detail below. The first grant was DUE-0241856 ($2,519,166, 2002-06), entitled *Multidisciplinary Systems Assurance Education.* As a Co-PI on this Federal Cyber Service Scholarship for Service Program grant, Liddy serves as mentor for Masters students in both the *Information Management* and *Telecommunication & Network Management* degree programs. Liddy has been able to involve the students actively in appropriate funded research projects underway at CNLP that address issues of information systems security and insider threats. Efforts have been established to enable these students to have summer internships with local companies whose expertise is in R & D on information systems security. The contribution to human resources is the main goal of this project and it is showing promising results.

The remaining four projects are funded by NSF's National Science Digital Library Program and involve research, implementation, and evaluation of NLP technology for automatic metadata generation for educational objects, most typically teachers' lesson plans and activity guides. The four grants are DUE-0085837 ($366,000, 2000-02) *Breaking the MetaData Generation Bottleneck*, DUE-0121543 ($475,000, 2001-03), *Standard Connection: Mapping Educational Objects to Content Standards*, DUE-0226312 ($374,938, 2002-04), *MetaTest: Evaluating the Quality & Utility of MetaData* and DUE-0435339 ($634,218, 2004-2006*), Computer-Assisted Standard Assignment & Alignment*. Two of the projects center around content standards, either their automatic assignment to resources or the automatic mapping amongst multiple national standards and the fifty state standards. Over the life of these four projects, Liddy and team have: 1) adapted their existing NLP methods and technology to the task of extracting from learning resources the values for the 23 metadata elements used for representing learning objects in digital libraries (15 Dublin Core + 8 GEM); 2) proven in end-user empirical evaluations that the metadata elements assigned automatically using NLP are equally good as those assigned by humans, and; 3) extended the metadata capability to map individual resources to the relevant content standards in Math and Science, key to standards-based education. Results were evaluated by experts in standards and classroom teachers. The fourth project is just underway and therefore there are no results as of yet. The grants have resulted in numerous publications [119-127,201]. Four PhD students and three Masters students have been active participants, learning both about the research and evaluation process and the wider field of digital libraries. They have presented the projects' findings jointly or singly and interacted substantively with this research community at relevant conferences.

## 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** 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

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

**9** 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

**10** 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

**11** 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

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

**13** Beißwenger, M. (2003) Bibliography of Chat Communications. Available from: http://www.chat-bibliography.de/, Accessed 17 February

**14** Belady, L.A. and Lehman, M.M. (1976) A model of large program development. *IBM Systems Journal*, 15(1), 225-252

**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** Bezroukov, N. (1999) A second look at the Cathedral and the Bazaar. *First Monday*, 4(12)

**18** Boehm, B. (1981) *Software Engineering Economics*: Prentice Hall

**19** Boehm, B., Clark, B., Horowitz, E., Madachy, R., Shelby, R. and Westland, C. (1995) Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*, 1, 57–94

**20** 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 Dis-*

*tributed 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

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

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

**23** 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

**24** 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

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

**26** 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

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

**28** 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

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

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

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

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

**33** Chidamber, S.R. and Kemerer, C.F. (1994) A metrics suite for object oriented design. *IEEE Transactions On Software Engineering*, 20(6), 476-493

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

**35** 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:

**36** 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

**37** 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

**38** 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:

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

**40** 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

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

**42** Crowston, K. and Howison, J. (In press) Hierarchy and Centralization in Free and Open Source Software team communications. *Knowledge, Technology & Policy*

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

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

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

**46** 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

**47** 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

**48** 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

**49** 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

**50** 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

**51** 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

**52** 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*

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

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

**55** 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:

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

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

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

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

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

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

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

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

**64** 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

**65** 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

**66** 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

**67** 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

**68** 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.

**69** 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

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

**71** Gall, H., Hajek, K. and Jazayeri, M. (1998) Detection of Logical Coupling Based on Product Release History. In *Proceedings of the International Conference on Software Maintenance (ICSM '98)*

**72** 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

**73** 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

**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) Prospering in dynamically-competitive environments: Organizational capability as knowledge integration. *Organizational Science*, 7(4), 375–387

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

**80** Graves, T.L. (1998) *Inferring Change Effort from Configuration Management Databases*

**81** 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

**82** 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

**83** Grishman, R. and Kittredge, R., eds (1986) *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, Lawrence Erlbaum

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

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

**86** 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

**87** 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

**88** 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

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

**90** 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

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

**92** 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

**93** 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

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

**95** 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

**96** 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:

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

**98** 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

**99** 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

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

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

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

**103** 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

**104** Kemerer, C.F. and Slaughter, S. (1999) An Empirical Approach to Studying Software Evolution. *IEEE Transactions on Software Engineering*, 25(4)

**105** 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

**106** 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

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

**108** 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

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

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

**111** 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:

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

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

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

**115** 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

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

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

**118** 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

**119** Liddy, E.D. (2001) Breaking the Metadata Generation Bottleneck. In *Joint Conference on Digital Libraries*, Roanoke, VA:

**120** Liddy, E.D. (2001) Data-Mining, MetaData, and Digital Libraries. In *DIMACS Workshop on Data Analysis and Digital Libraries*, Rutgers University:

**121** Liddy, E.D. (2003) Automating and Evaluating Automatic Metadata Generation. In *Search Engine Conference*, Boston, MA:

**122** Liddy, E.D. (2003) Automating and Evaluating Metadata Generation. In *Libraries in the Digital Age Conference*, Dubrovnik, Croatia.:

**123** Liddy, E.D. (2003) Automating and Evaluating Metadata Generation. In *InfoToday 2003 Conference*, New York, NY:

**124** Liddy, E.D. (2003) Natural Language Processing for Text Extraction Applications. Keynote Speaker. In *Thomson 8th Annual Text Summit*, Minneapolis, MN

**125** Liddy, E.D. and Finneran, C. (2003) Developing and Evaluating Metadata for Improved Information Access. In *NSF-NSDL Annual Meeting*, Washington, DC:

**126** Liddy, E.D. and Finneran, C. (2003) MetaTest: Three-Way Evaluation of Automatic Metadata Generation. In *Joint IMLS/NSDL Conference*

**127** Liddy, E.D., Gay, G., Harwell, S. and Finneran, T. (2002) A Modest (Metadata) Proposal. In *Joint Conference on Digital Libraries*, Portland, OR:

**128** Liddy, E.D., Jorgensen, C.L., Sibert, E.E. and Yu, E.S. (1991) Sublanguage grammar in natural language processing. In *Proceedings of RIAO '91 Conference*, Barcelona:

**129** Liddy, E.D., Jorgensen, C.L., Sibert, E.E. and Yu, E.S. (1993) A sublanguage approach to Natural Language Processing for an expert system. *Information Processing & Management*, 29(5), 633–645

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

**131** 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

**132** 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

**133** 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

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

**135** 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

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

**137** 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

**138** Miles, M.B. and Huberman, A.M. (1994) *Qualitative Data Analysis: An Expanded Source-book*, Thousand Oaks: Sage Publications

**139** 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

**140** 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

**141** 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

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

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

**144** 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

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

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

**147** 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

**148** 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

**149** 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

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

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

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

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

**154** 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

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

**156** Perry, D.E., Staudenmayer, N.A. and Volta, L.G. (1994) People, organizations, and process improvement. *IEEE Software*, 11(4), 36–45

**157** 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

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

**159** Rambow, O., Shrestha, L., Chen, J. and Laurdisen, C. (2004) Summarizing Email Threads. In *Proceedings of HLT-NAACL*, pp. 105–108

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

**161** Raymond, E.S. (1998) Homesteading the noosphere.

**162** 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

**163** 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

**164** 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

**165** Sager, N., Friedman, C. and Lyman, M.S. (1987) *Medical Language Processing: Computer Management of Narrative Data*, Reading, Mass: Addison-Wesley

**166** 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

**167** 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

**168** 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

**169** 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

**170** 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

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

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

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

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

**175** 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

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

**177** 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

**178** 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

**179** 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

**180** 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

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

**182** Tuomi, I. (2002) Evolution of the Linux Credits File: Methodological Challenges and Reference Data for Open Source Research. Available from: http://www.jrc.es/~tuomiil/articles /EvolutionOfTheLinuxCreditsFile.pdf, Accessed 15 November

**183** 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

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

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

**186** 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

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

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

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

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

**191** 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

**192** 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

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

**194** 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

**195** 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

**196** 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

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

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

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

**200** 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:

**201** Yilmazel, O., Finneran, C.M. and Liddy, E.D. (2004) MetaExtract: An NLP System to Automatically Assign Metadata. In *Proceedings of the 2004 Joint Conference on Digital Libraries*

**202** 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