DHB: Investigating socialization of new members into self-organizing technology-supported distributed teams

Kevin Crowston, PI, Syracuse University School of Information Studies

Project Summary

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. These distances are particularly problematic for new members seeking to join the teams. But as yet, research and practitioner communities know little about the dynamics of socialization in distributed teams, which our literature review suggests are likely to be substantially different from those in others kinds of organizations. These dynamics are particularly challenging when teams have the autonomy or responsibility to self-organize (e.g., in teams that span multiple formal organizations). The goal of our study is to better understand the cognitive and social structures that underlie changes in individual and team behaviors in these teams as this mode of work is becoming increasingly more common. Our study addresses the general research question**: What are the dynamics by which new members are socialized into self-organizing technology-supported distributed teams?**

To study the dynamics of self-organizing distributed teams, specifically new member socialization, we propose a multi-disciplinary and inter-disciplinary study that integrates the analysis of multiple sources of data using multiple research methods. We first review the literature on socialization in conventional organizations and in FLOSS projects to develop a theoretical framework to guide our study. We will use a combination of human coding, 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 program source code are readily available for study.

Expected intellectual merits

The proposed study will have conceptual, methodological as well as practical contributions. Developing an integrated theoretical framework to understand the socialization of new members into of a distributed team will be a contribution to the study of distributed teams, an increasingly common mode of work. The project will advance knowledge and understanding of FLOSS development specifically 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 benefit society by suggesting ways to strengthen distributed FLOSS teams, an increasingly important approach to software development. We are particularly interested in why FLOSS teams seem to attract so few women. 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 infrastructure for research, we plan to make our tools and data available to other researchers. Finally, the project will promote teaching, training, and learning by providing an opportunity for students to work on research teams, utilize their competencies and develop new skills in data collection and analysis.

DHB: Investigating socialization of new members into self-organizing technology-supported distributed teams

Project Description

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. These distances are particularly problematic for new members seeking to join the teams. But as yet, research and practitioner communities know little about the dynamics of socialization in distributed teams, which our literature review suggests are likely to be substantially different from those in others kinds of organizations. These dynamics are particularly challenging when teams have the autonomy or responsibility to self-organize (e.g., in teams that span multiple formal 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 behaviors in these teams. Our study addresses the general research question**: What are the dynamics by which new members are socialized into self-organizing technology-supported distributed teams?**

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, 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).

The research literature on software development and on distributed work emphasizes the difficulties of distributed software development, but the apparent success of FLOSS development presents an intriguing counter-example. Characterized by a globally distributed developer force and a rapid and reliable software development process, effective FLOSS development teams somehow profit from the advantages and overcome the challenges of distributed work [3]. Traditional organizations have taken note of these successes and have sought ways of leveraging FLOSS methods for their own distributed teams. Thus, while in many ways unique, the distributed and self-organizing nature of FLOSS teams represents a mode of work that is

1 FLOSS software is usually available without charge (captured in a phrase commonly used in the community: "free as in free 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 (captured as "free as in free speech", thus "libre"). We have chosen to use the acronym FLOSS rather than the more common OSS to acknowledge this dual meaning.

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 [61]. FLOSS is an 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 is heavily dependent on FLOSS tools. These systems are an integral part of the infrastructure of modern society, making it critical to understand more fully how they are developed. Furthermore, FLOSS is an increasingly important venue for students learning about software development. However, as Scacchi [128] 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". Indeed, the term FLOSS includes groups with a wide diversity of practices, with varying degrees of effectiveness, but the dimensions of this space are still unclear. Of particular concern is the extremely low percentage of women participating in these projects, which we suspect is related to their socialization practices.

Key to our interest in these teams is the fact that most FLOSS projects are developed by dynamic, self-organizing, distributed teams comprising professionals, users [143-145] 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) [123, 149]. The teams have a high isolation index [115] 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.

Another important feature of the FLOSS development process is that developers contribute to projects as volunteers, the majority without being paid at all; others are paid by their employers, but not by the project. As a result, recruiting and retaining new contributors is a critical success factor for a FLOSS project. These features make FLOSS teams extreme examples of self-organizing distributed teams, but they are not inconsistent with what many organizations are facing in recruiting and motivating professionals and in developing distributed teams. As Peter Drucker put it, "increasingly employees are going to be volunteers, because a knowledge worker has mobility and can go pretty much every place, and knows it… Businesses will have to learn to treat knowledge workers as volunteers" [23]. The combination of these two features implies that socialization of new members is critical to the success of these teams, even though conventional approaches to socialization are difficult or impossible to apply.

Multidisciplinarity and lnterdisciplinarity

To study the dynamics of socialization of new members into self-organizing distributed teams, we propose a multi-disciplinary and inter-disciplinary study that integrates the analysis of multiple sources of data using multiple research methods. We will use a combination of human

coding, 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. To accomplish this study, we have assembled a multi-disciplinary team, bringing together researchers with diverse backgrounds, but with experience working together in a multi-disciplinary school.

Fit to Human and Social Dynamics

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, but which fits the call for research on human and social dynamics. 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 qualitative aspects of email messages will drive 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 theories of new member socialization [6, 111, 141] as an organizing framework. In section 3, we present the study design, with details of the data collection and analysis plans. In that same 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.

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., 113], advances in information and communication technologies have been crucial enablers for recent developments of this organizational form [2] and as a result, distributed teams are becoming more popular [105]. Distributed teams seem particularly attractive for software development because the code can be shared via the same systems used to support team interactions [112, 127]. While distributed teams have many potential benefits, distributed workers face many real challenges. Watson-Manheim, Chudoba & Crowston [148] suggested 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 [140], or that produces unintended

information filtering [45] or misunderstandings [5]. These interpretative difficulties, in turn, make it hard for team members to develop shared mental models of the developing project [44, 58]. 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 [103]. These issues are particularly critical for new member socialization.

The presence of discontinuities seems likely to be particularly problematic for software developers [140], hence our interest in distributed software development. Numerous studies of the social aspects of software development teams [43, 83, 126, 140, 147] conclude that large system development requires knowledge from many domains, which is thinly spread among developers [43]. As a result, large projects require a high degree of knowledge integration and the coordinated efforts of multiple developers [19]. More effort is required for interaction when participants are distant and unfamiliar with each others' work [116, 131]. The additional effort required for distributed work often translates into delays in software release compared to traditional face-to-face teams [76, 107]. 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 [75].

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 [20], again pointing to the importance of new member socialization. Research has shown the importance of formal and informal coordination mechanisms and information sharing [147] 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 [64, 74, 84, 86, 89]. Successful distributed teams share knowledge and information and create new practices to meet the task-related and social needs of the members [124]. However, the dynamics of socialization for distributed teams are still open topics for research [117].

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., 46, 88, 95]. Second, various explanations have been proposed for why individuals decide to contribute to projects without pay [e.g., 15, 62, 70, 78, 104]. These authors have mentioned factors such as increasing the usefulness of the software [71], personal interest [71], ideological commitment, development of skills [99] with potential career impact [71] or enhancement of reputation [104]. Finally, a few authors have investigated the processes of FLOSS development [e.g., 123, 133]; those processes are the focus of this proposal.

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., [134]). 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 [37, 69, 94, 120, 121,

137, 138]. 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 [109, 142], 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 [90]. 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.

Only a few studies have examined socialization of members into FLOSS teams. von Krogh et al [146] studied socialization in the Freenet project, focusing on discovering 'joining scripts,' which they describe as the "level and type of activity a joiner goes through to become a member of the developer community" (p 1227). They found that eventual joiners were more likely to accompany their first contact with the project with incremental code (bug-fix or small feature addition) and to continue their interaction at a substantially higher level of activity focused at a technical level. They contrasted this pattern of behavior with two other approaches, the 'resume' approach (listing skills and asking for work) and the 'visioneer' approach (raising major architectural issues or "great ideas," without providing code), which were far less likely to result in eventual developer status. Further, they found that newcomers were far more likely to contribute in areas such as the build system or new clients, rather than in areas with higher 'contribution barriers' such as the encryption or security systems. Duchenaut [56] studied socialization in the Python project and found that participants who join the centre of a project act in a way that exposes more of the 'network' to them, so they come to understand the relationships between people and code. Largely through action, in the form of code or detailed discussions of code, they built legitimacy and "enrolled allies" for their evolution towards the core of the project. These studies, while suggestive, are case studies of only a few individuals in two projects. They suggest possible trajectories for a new member, but provide few details about the dynamics of the socialization process, which is our objective.

2. Conceptual development

In this section we develop the conceptual framework for our study. For this project, we have chosen to analyze individuals who are involved with a particular FLOSS project as comprising members of a non-traditional organization. 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 (though not the process by which they do so). Other researchers have described them as communities of practice, which is a helpful lens for studying how knowledge and practices are shared (as we discuss below). Still others have referred to FLOSS project members as teams. 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 [118]), members share common practices, but are individually responsible for their own tasks. However, because our study focuses on new member socialization, we are viewing the projects as organizational entities that have a goal of developing a product, that have a user base to satisfy, and that share a common social identity as members of the project. Project members are interdependent in terms of tasks and roles and core

members know and acknowledge each other's contributions. Differentiation and specialization of roles evolves over time. Together, core members and others involved in the project are embedded in a larger social system that comprises the project as a whole. Treating the FLOSS project as an organization—albeit not a traditional bricks and mortar organization—allows us to tap the rich vein of social science literature on new member socialization.

In the context of traditional organizations, new member socialization has been a persistent topic of interest for decades because of its influences on organizational success [129]. Among organizational scholars, socialization is usually defined as the interrelated processes by which a new entrant to an existing organization transforms from outsider to organizational insider [60]. Generally speaking, this transformation comprises two simultaneous sets of processes. The most visible socialization processes involve learning the tasks associated with a particular organizational role, the knowledge and skills needed to fulfill that role and the privileges and responsibilities that accompany it. These role-related socializations may include training, orientation, practice, receipt of mentoring and apprenticeship, as well as rituals of testing and initiation. Less visible but equally important are the processes of socialization unrelated to a specific role or set of job tasks. These processes include learning the norms for behavior in the organization, understanding the informal power hierarchy, and developing a network of colleagues that provide non-task support [7]. Membership in the in- and out-groups may be made visible through adoption of distinctive in-group/out-group markers, which again are learned as part of socialization.

Research evidence indicates that successful socialization of new members can lead to a variety of benefits for both the new member and the organization [101, 119]. Yet Van Maanen and Schein caution that not all socialization is beneficial to the individual, the organization, or both [141]. Organizational forms that persist and that are communicated to or imposed upon new members have their roots in an organizational history that may be irrelevant to the current climate. For example, Van Maanen and Schein describe so-called social filters that have their greatest impact on newcomers as they make transitions across organizational boundaries. Social filters are defined as the human interaction and communication processes by which an individual becomes accepted (or fails to become accepted) as a new member of a group. Sometimes these filters inhibit or prevent successful transitions across organizational boundaries. Note that in this view, the transition between outsider and insider is merely the first of several possible transitions. As individuals take on new functional roles or as individuals change positions in the social power hierarchy within the organization, they become newcomers again in their new roles and must to some degree adapt to and learn the new situation.

Based on results from our previous NSF award in this area (HSD Grant 05-27457, reviewed below), it has become evident that, despite an overall quite flat formal organizational structure, FLOSS projects contain at least four distinct and nested membership levels. Working from the inside out, each FLOSS project contains a core developer group whose roles vary with the maturity of the project but who have primary responsibility for project strategy, architecture, and, importantly, membership. At the next level outward, there is a secondary developer community, roughly speaking an order of magnitude larger than the core group, who contribute to the project in modest ways (e.g., bug fixing). At the third level, there is a large collection of power users who have substantial involvement on project matters unrelated to the creation of code (e.g.,

documentation, bug reporting). Finally, at the outermost level there is a user population, who we consider interested outsiders in terms of the socialization frameworks we examined.

Newcomers who eventually join the core developer group have typically progressed across the boundaries that separate each of these four groups through a series of role transitions. In much of the prior thinking about socialization in FLOSS, the outsider to insider transition has been narrowly defined as the process of gaining access to the version control system (i.e., the ability to submit code directly to the current version under development), which is to say, crossing the boundary to core developer. We believe that new member socialization begins earlier and ends later than this, and that current research efforts have failed to take into account the mutuality of the socialization process, as we will discuss. Taking each level in turn:

- There are essentially no social barriers to joining the user population, and as mentioned above, we are asserting that users should be construed as outsiders to the project for purposes of theorizing about newcomer socialization.
- Transitioning from user to power user also has a low barrier to entry. However, social filters may act in a dysfunctional way by discouraging newcomers from becoming fully socialized into further participation in the project. Here we are explicitly drawing an analogy between the traditional organization and the FLOSS team, with the understanding that there are important differences that will be described below. So-called "inclusionary" filters may operate to retard the progress of an individual towards an accepted role within the power user group [100]. More specifically, before obtaining acceptance as an insider, a prospective power user may have to submit a set of contributions to the FLOSS project that are consensually viewed by the core and secondary developers as valuable to the project, an example of testing and initiation.
- These inclusionary filters operate even more powerfully in the transition from power user to secondary developer. Because the widespread use of version control allows members of the core developer group to regulate the acceptance of all code submissions to the project, a prospective secondary developer's opportunity to join this group depends upon the explicit acceptance of a contribution to the project by members of the core group. This point of acceptance is the juncture where socialization processes become highly relevant to the success of the FLOSS project. Core developers have a strong interest in regulating the quality of code submissions and to filter out inappropriate contributors, but they also have a strong interest in successful adoption and enculturation of new members.
- Finally, individuals who wish to make the transition into the secondary developers group or from the secondary group into the core have a strong interest in obtaining the social acceptance of the core group and the existing core group members, in having an additional effective contributor.

At this point though, the conventional theories of newcomer socialization break down as a result of the distinctions between traditional organizations and FLOSS project groups. When a traditional organization selects a new member and the new member accepts membership in the organization, the new member is then obliged to follow both formal and informal socialization activities [6]. In essence, most of the formal socialization activities and many of the informal socialization activities are non-discretionary once the new member has accepted a role in the traditional organization. In contrast, there is a strong mutuality to the process of new member

socialization in the context of an open source project. There is no point in the socialization process where either the new member or the existing membership is obliged to accept overtures from the other. The process of socializing the new member is in effect much more voluntary than in a traditional organization. New members may withdraw from the socialization process at any time, particularly if the existing members fail to provide a workable and valued role to the new member that can help cement his or her social identity in the group.

One highly noticeable side effect of this voluntary process is its informality. Few open source projects provide any formal socialization procedures to new members [16, 55]. In addition to this important structural difference, there are several logistical differences that make traditional organizational socialization distinct from a FLOSS project. For example, in the traditional organization—even in those organizations that are highly distributed and virtual—there is generally an initial period of intensive face-to-face contact between the new member and his or her coworkers, while such face-to-face meetings are rare for FLOSS projects and tend to happen much later in the socialization process [35].

Together, these elements suggest that traditional theories of new member socialization (that is, those developed for the context of traditional organizations) may yet have applicability to the open source project situation, albeit in modified form. Thus, an important scientific goal of this research is to assess the usability of new member socialization theories to socialization processes in open source projects [6, 110, 111]. Prior research of software development emphasizes the importance of new project members learning a wide variety of skills and knowledge to be effective in their new roles: details of the project being developed, status and competencies of other participants, formal and informal coordination mechanisms and so on. Distributed team members also need to learn effective modes of interaction using computer-supported communications tools. Research on new member socialization describes how new members are taught tasks associated with a new role, knowledge and skills needed to fulfill that role, as well as norms for behavior in the organization and the informal power hierarchy, and how they develop a support network in the organization. However, many of the mechanisms observed in traditional groups to support this learning, such as mentoring or face-to-face meetings, seem difficult or impossible to apply in distributed groups, even though several points of socialization can be identified. Our study addresses this disconnect by describing the dynamics by which new members are socialized into self-organizing technology-supported distributed teams.

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 new member socialization among FLOSS developers, we develop an innovative multi-disciplinary approach to our research. For each project element, we will draw on multiple sources of data: developer interactions, project and developer demographics, project plans and procedures, interviews with developers, and the source code. The data will be analyzed using social network analysis (SNA), and content analysis using human and Natural Language Processing (NLP) techniques to reveal the dynamics of new member socialization.

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 1. In the first phase (roughly year 1), we will examine project transcripts manually for evidence of the dynamics of new member socialization 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 and apply 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. In the remainder of this section, we will provide more detail on our proposed plan of study, 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 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 significant new member socialization activities (as opposed to single person development efforts [91]), we will choose only projects with more than seven core developers [72]. We will include both mature and newly forming teams, though a significant advantage of studying FLOSS teams is the ability to collect data across 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).

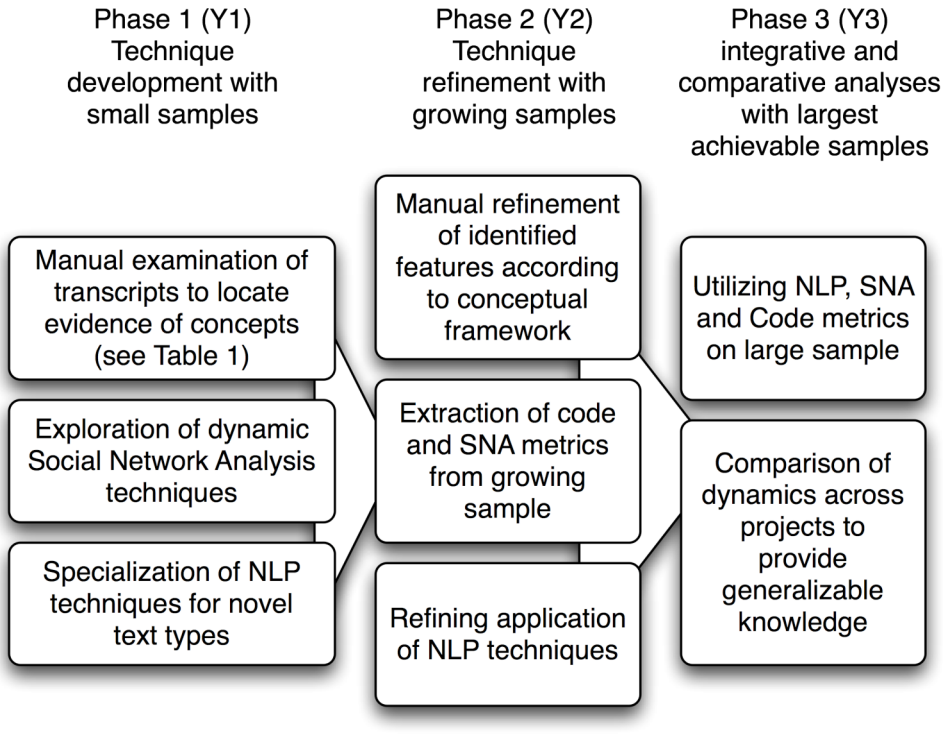

Figure 1. Phases of research plan.

Step two: Charting flows of actions. In this step, we analyze the actions of team members within a particular time period. We will extract interactions from email logs and other forums and identify 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 new member socialization dynamics in the teams. More specifically we will uncover the mechanisms of behavior and communication through which members progress from outsider (optionally) to power user, to secondary developer, to core developer with the recognition that these prototypical transitions may oversimplify the actual dynamics and with the openness to adapt our theory to that empirical reality. 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. We are particularly interested in identifying points at which women may be differentially affected by the socialization process, causing them to be filtered or self-selected out. Lastly, we will study the dynamics by which norms evolve for membership and other aspects of social identity.

Step four: Linking components of socialization to other project dynamics and outcomes. To accomplish this step, we will triangulate evidence gathered from multiple sources of evidence about the teams. For example, the implications of socialization will be assessed by linking them to changes in the source code, team outputs and measures of project growth.

Data collection

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

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

Concept	**Constructs**	**Data sources and analysis**
Social identity	Roles with distinctive tasks, knowledge, skills, privileges and responsibilities	Developer demographics, content analysis of interaction logs, social network analysis, code analysis
Insider knowledge	Norms and power structure In-group/out-group markers	Content analysis of interaction logs, interviews
Social network	Support network of help and advice	Social network analysis
Role transitions	Joining scripts; implicit and explicit membership requests Inclusionary, functional and hierarchical filters	Developer demographics, content analysis of interaction logs, social network analysis, interviews

Socialization processes	Testing and initiation	Content analysis of interaction logs, interviews, code analysis
Individual effectiveness	Contribution to project outputs	Code analysis

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 (encryption) key signatures, we can identify meetings between developers [114], which will suggest past opportunities for face-to-face interaction.

Interaction logs. The most voluminous source of data will be collected from archives of computer mediated communication tools used to support the teams' interactions for FLOSS development work [76, 93]. These data are useful because they are unobtrusive measures of the team's behaviors[2] [150]. Mailing list archives will be a primary source of interaction data that illuminates the 'scripts' for the analysis of dynamics [11], as email is one of the primary tools used to support team communication, learning and socialization [92]. Such archives contain a huge amount of data (e.g., the Linux kernel list receives 5k-7k messages per month, the Apache httpd list receives an average of 40 messages a day). From mailing lists, we will extract the date, identifiers of sender and individual recipients, 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 [57] found relations between level, pattern and content of messages and team performance. In addition to email, we will examine logs from other interaction tools, such as chat sessions, bug reports and features request archives.

Interviews. Despite the depth and richness of interaction logs, there are some aspects and activities of each open source project that occur behind the scenes, outside the view of archived messages. Of particular importance with respect to new member socialization are the tacit rules and norms used by members as they judge requests by nonmembers. Likewise, some of the thinking and motivations of nonmembers are never made explicit in the context of the interaction logs. Face-to-face, telephone, and e-mail interviews provide a means of obtaining these otherwise invisible data.

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

[2] Because these communications are posted with the explicit intent that they be public, the Syracuse University Institutional Review Board considers them public behaviors that do not require informed consent before study. Nevertheless, we plan to follow the ethical recommendations of Association of Internet Researchers [59] in designing our study and publishing our results.

code in order to understand the software structure and the role of individual developers [63, 65, 108]. Raw software code poses numerous challenges to interpretation [135]. For example, not all projects assign authorship in the CVS tree. We plan to leverage our analysis with work being carried out by other researchers in order to deal with these challenges [e.g., 87].

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 shown in Figure 2. 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 activities involved in new member socialization. The final stage is to develop process maps that document the individual and collective actions and the dynamics of change that address our research questions. These results will show the new socialization processes in each project. 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.

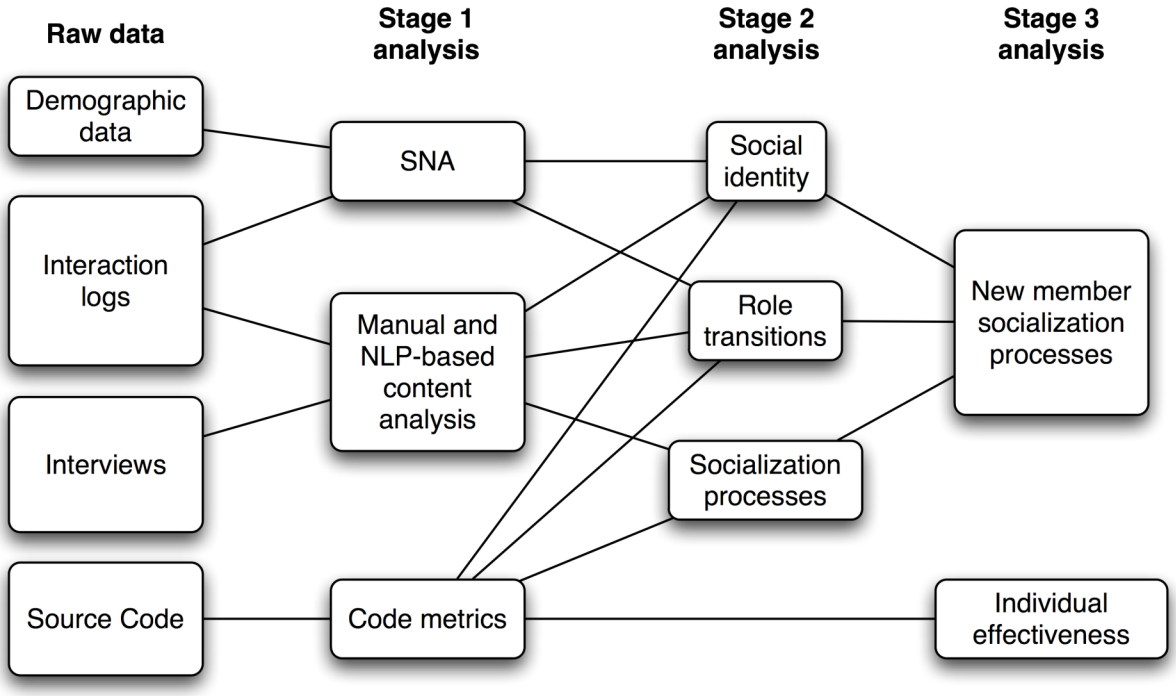

Figure 2. Data analysis, from raw data to team dynamics and outcomes.

Content analysis. Content analysis of computer-mediated communication (CMC) has been an active area of research [12, 77]. 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 [106], 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 new member socialization. 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. [136] 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 [122]. To effectively meet the challenge of understanding these stylistically diverse and grammatically inconsistent texts, the 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 [66, 97, 98, 125] 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 [98]. 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.

In the current NSF-funded project, *DHB: Investigating the Dynamics of Free/Libre Open Source Software Development Teams* (HSD 05–27457), the NLP technology has been customized to automatically code email text for evidence that a decision has been announced, and some work has been done to automatically code 'decision triggers', or issues that have opened up the opportunity for a decision to be made. The automatic coding is based on a gold standard of manually coded data. Six FLOSS projects were manually coded, and the email data was separately run through the automatic coding and evaluated for recall (the proportion of manually coded data that was automatically coded), precision (the proportion of automatically coded data that was in agreement with the manually coded data), and utility (a subjective measure that indicates the proportion of automatically coded data that is judged useful to understanding the decision processes within the email texts.) Overall recall is 76%, precision is 56%, and utility is 75%. This shows promise in automatic coding for content analysis, but needs testing on a larger scale, and with multiple code types. The proposed research is designed to further extend and evaluate this technology.

To accomplish automatic coding objectives, the first task is to prepare all of the data for text processing through a pre-processing phase that will capture metadata for each message or data unit and structure each into an XML representation. Several pre-processors may need to be created for the varying discourse structures found in the data to be collected. After manual analysis has begun, and the aspects of socialization, conversation and recapitulation related to shared mental models, and role definitions have been codified, the NLP rule sets will be refined and extended to enable automatic identification and extraction of text that reflects and instantiates the target codes. This will be an iterative process of rule building, testing and analysis that will extend beyond phase 1 through the subsequent phases of analysis, whereby the growing understanding of the social structures, norms and role of each individual will provide further feedback, insight and guidance for additional refinements to the NLP technology.

Social network analysis (SNA). Social network analysis will be used to analyze patterns of interactions (e.g., who responds to whose email). Madey, Freeh & Tynan [102] applied SNA to

connections between projects, but not within projects. Ducheneaut [54] 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 [30] and most projects were quite hierarchical [32], similar to the results of Ahuja & Carley [1]. By analyzing the social network of a project, we will assess each individual's centrality to the project and relations to other developers, thus revealing the processes and results of socialization. We can also determine the project's level of hierarchy, which seems to mediate the effect of role and status on individual performance within virtual teams [2]. 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.

Software source code analysis. In analyzing the teams' output, we will focus on the program software source code (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 as new members join and what it actually does. Code complexity may make it more difficult for new members to join or channel their efforts in particular ways, as well as being a result of contributions from less experienced developers. 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 [17, 18] and the "CK" suite of metrics [22].

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 [13], 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 [85].

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. First, we will replicate previous work on *joining scripts* (see Table 1) by examining requests that new members make to members of the core developer team. In examining the responses to such requests, we will seek evidence of the inclusionary, functional, and hierarchical filters that traditional and new member socialization theory suggests will be used to regulate the boundaries between different membership categories. Expanding our lens, we will focus more broadly on evidence of the dynamics of *socialization*. In particular, beyond the filtering activity mentioned above, most socialization processes include elements of testing—where the fitness of an outsider is examined—and initiation, in which new members receive information and instruction on the rules and norms that govern behavior in the group. These processes are reflected in various markers that indicate the status of an individual with

respect to the group. The most obvious and pronounced marker of in-group status in an open source project is the ability to post new or modified code directly to the version control system.

Most theories of socialization account for issues of social *identity*. Newcomers to an organization gradually adopt a social identity that meshes with the existing culture and structures of the organization. The social identity, in turn, usually reflects a composite of different roles assigned or undertaken by the member. 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 block-models 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.

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 socialization dynamics of the FLOSS teams, with large volumes of data automatically coded for analysis. 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 [139] 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 relating to new member socialization. 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 different roles 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. Differences in the quality or quantity of contributions can be correlated back to differences in the formality, length, and extent of 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 using

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

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 and summer support for 2 PIs and calendar support for an additional PI. The PI, Crowston has a PhD in the field of Management and publishes mostly in the Information Systems area (Crowston also works in Organizational Communications). The first co-PI, Stanton, has a PhD in organizational psychology, an undergraduate degree in computer science, and more than a decade of experience as a professional software developer. The final co-PI, Diekema, has a PhD in Information Transfer and works in the field of Natural Language Processing.

All three PIs, Kevin Crowston, Stanton and Diekema, will work 0.5 months each on the project. Crowston and Stanton will work during the summer on project management and research design and devote time during the academic year to project management and oversight as part of their regular responsibilities. As a Research Professor, Diekema will work during the calendar year, contributing to project management, research design and oversight. 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, Stanton will oversee the interviewing and initial coding of developer transcripts and Diekema 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.

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 have developed a conceptual framework and a research plan to investigate new member socialization practices within distributed FLOSS development teams. To answer our research question (What are the dynamics by which new members are socialized into self-organizing technology-supported distributed teams?), we plan to conduct a longitudinal in-depth study identifying and comparing the evolution of distributed teams of FLOSS developers. We will study how these distributed groups develop establish tacit filters for membership, regulate roles and social identities, and set norms and rules that distinguish the roles of insiders and outsiders. We will also examine how new members enact joining scripts and negotiate the boundary between out-group and in-group status, as well as how they acculturate to newly obtained membership status. Finally, we will relate new member socialization practices to the outcomes of each open source project including measures of growth and quality.

Expected intellectual merits

The project will contribute to advancing knowledge and understanding of self-organizing distributed teams by identifying and examining the dynamics of new member socialization in 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 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 methodological 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 been extensively 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 uncovering the dynamics of teams in FLOSS development, an increasingly important approach to software development. The study will also shed light on dynamics of socialization for distributed work teams in general, which will be valuable for managers who intend to implement this organizational form. Understanding the dynamics of these teams can provide guidelines for socialization 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 [14, 21, 84, 89]. For the potential of distributed teams to be fully realized, research is needed on the dynamics of 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 [4] note, in these teams, "developers collectively acquire and develop new skills and experiences". A particular question we hope to address here is reasons for the low level of participation by women in FLOSS projects.

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 will propose a book to a publisher at the close of the project. We also plan to disseminate results directly to practitioners through interactions with our advisory board and with developers, e.g., at FLOSS conferences, a practice that has proven very beneficial on our previous grants. 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. 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.

Results from prior NSF funding

One of the PIs for this grant, Crowston, has been funded by several additional NSF grants within the past 48 months. The grants most relevant to the current proposal is HSD 05–27457 ($684,882, 2005–2008, with R. Heckman, E. Liddy and N. McCracken), *Investigating the Dynamics of Free/Libre Open Source Software Development Teams*, IIS 04–14468 ($327,026, 2004–2006) and SGER IIS 03–41475 ($12,052, 2003–2004), both entitled *Effective work practices for Open Source Software development* and CNS Grant 07-08437 ($200,000, 2007–2010, with M. Conklin, Elon University), for *Collaborative Research: CRI: CRD: Data and analysis archive for research on Free and Open Source Software and its development*. The first three of these grants have supported a study of the evolution of effective work practices for distributed groups, specifically, for teams of free/libre open source software (FLOSS) developers. The funding enabled travel to conferences (e.g., *ApacheCon* and *OSCon*) to observe and interview FLOSS developers and to present preliminary results, and for the purchase of data analysis software and equipment. The final grant is supporting the development of cyber-infrastructure to support the FLOSS research community more broadly. We plan to leverage this

investment in supporting the proposed work. Overall, this work has resulted in nine journal papers [30, 31, 33, 35-37, 39, 41, 79], multiple conference papers [e.g., 25, 28, 34, 38, 40, 42, 73, 81, 82, 96, 130] and workshop presentations [24, 26, 27, 29, 80]. These grants have supported a total of six PhD students; several others have been involved in specific aspects of the work. The HSD grant included a component applying NLP techniques to analyze large corpora of email (as noted above) and provided significant experience working in an interdisciplinary team.

Both co-PIs have also received NSF support. The first, Diekema has been funded by NSF's National Science Digital Library Program in projects that involve research, implementation, and evaluation of NLP technology for automatic metadata generation for teachers' lesson plans, specifically educational standards. The grants are DUE-0121543 ($475,000; 2001-03) *Standard Connection: Mapping Educational Objects to Content Standards*; where Diekema was Project Manager; and as Co-PI on DUE-0435339 ($634,218; 2004-06 plus an extension $74,911; 2007-2008) *Computer-Assisted Standard Assignment & Alignment*. These projects revolve around educational content standards, either their automatic assignment to resources or the automatic mapping amongst multiple national and state standards. Over the life of the projects, Diekema and team have: 1) adapted their existing NLP methods and technology to the task of establishing a connection between lesson plan content and educational standard content; and 2) created a web-based content assignment and alignment system that assist catalogers in mapping individual resources to the relevant content standards in Math and Science, essential for standards-based education. Results received highly positive evaluations by classroom teachers and experts in standards and the resulting systems are being used by catalogers today in large Digital Libraries such as Thinkfinity.org. The grants have resulted in numerous publications [51, 53, 151] and presentations [9, 47-53].

Stanton's prior NSF work includes an award entitled, "Culture Clash! The Adverse Effects of IT Occupational Subculture on Formative Work Experiences of IT Students" (CNS-0420434; 2004-2007, $299,000). In this research Dr. Stanton and his research team have uncovered core aspects of the occupational subculture (in U.S. organizations) in the information technology field that affect the beliefs and actions of women and minority students in the field. Adverse effects of such cultural characteristics as "pervasiveness" tend to make the information technology field less attractive to female and minority students, and sometimes influence them to abandon their information-related majors [67, 68, 132].

References

[1] M. K. Ahuja and K. Carley, "Network structure in virtual organizations," *Journal of Computer-Mediated Communication*, vol. 3, no. 4, 1998.

[2] M. K. Ahuja, K. Carley, and D. F. Galletta, "Individual performance in distributed design groups: An empirical study," in *Proceedings of SIGMIS Computer Personnel Research Conference*, San Francisco, CA, 1997, pp. 160–170.

[3] K. Alho and R. Sulonen, "Supporting virtual software projects on the Web," presented at Workshop on Coordinating Distributed Software Development Projects, 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '98), Palo Alto, CA, USA, June 17-19 1998.

[4] J. Arent and J. Nørbjerg, "Software Process Improvement as Organizational Knowledge Creation: A Multiple Case Analysis," in *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS-33)*, Wailea, Maui, HI 2000, pp. 11 pages.

[5] D. J. Armstrong and P. Cole, "Managing distance and differences in geographically distributed work groups," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 167–186.

[6] B. E. Ashforth and F. Mael, "Social Identity Theory and the Organization," *Organizational Identity: A Reader*, 2004.

[7] B. E. Ashforth and A. M. Saks, "Socialization Tactics: Longitudinal Effects on Newcomer Adjustment," *The Academy of Management Journal*, vol. 39, no. 1, pp. 149-178, 1996.

[8] U. Asklund and L. Bendix, "Configuration Management for Open Source Software," Department of Computer Science, Aalborg University, Aalborg, Denmark R-01-5005, 2001.

[9] J. A. Bailey, H. Devaul, A. R. Diekema, J. Ostwald, and J. Weatherly, "Educational Standard Correlation Services for Digital Libraries," in *Proceedings of National Science Digital Library Annual Meeting*, Arlington, VA, 7 Nov 2007.

[10] D. Bandow, "Geographically distributed work groups and IT: A case study of working relationships and IS professionals," in *Proceedings of ACM SIGMIS Computer Personnel Research Conference* 1997, pp. 87–92.

[11] S. R. Barley and P. S. Tolbert, "Institutionalization and structuration: Studying the links between action and institution," *Organization Studies*, vol. 18, no. 1, pp. 93–117, 1997.

[12] M. Beißwenger, "Bibliography of Chat Communications," [Online document], 2003, [cited 17 February 2004], Available HTTP: http://www.chat-bibliography.de/

[13] L. A. Belady and M. M. Lehman, "A model of large program development," *IBM*

Systems Journal, vol. 15, no. 1, pp. 225-252, 1976.

[14] F. Bélanger and R. Collins, "Distributed Work Arrangements: A Research Framework," *The Information Society*, vol. 14, no. 2, pp. 137–152, 1998.

[15] J. Bessen, "Open Source Software: Free Provision of Complex Public Goods," Research on Innovation July 2002.

[16] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu, "Open Borders? Immigration in Open Source Projects," *Proceedings of the Fourth International Workshop on Mining Software Repositories*, 2007.

[17] B. Boehm, *Software Engineering Economics*: Prentice Hall, 1981.

[18] B. Boehm, B. Clark, E. Horowitz, R. Madachy, R. Shelby, and C. Westland, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Engineering*, vol. 1, pp. 57–94, 1995.

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

[20] B. Butler, L. Sproull, S. Kiesler, and R. Kraut, "Community effort in online groups: Who does the work and why?," in *Leadership at a Distance*, S. Weisband and L. Atwater, Eds. Mahwah, NJ: Lawrence Erlbaum, 2002.

[21] E. Carmel and R. Agarwal, "Tactical approaches for alleviating distance in global software development," *IEEE Software*, no. March/April, pp. 22–29, 2001.

[22] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions On Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.

[23] J. Colllins and P. Drucker, "A Conversation between Jim Collins and Peter Drucker," in *Drucker Foundation News*, vol. 7, 1999, pp. 4–5.

[24] M. Conklin, J. Howison, and K. Crowston, "Collaboration Using OSSmole: A repository of FLOSS data and analyses," presented at Symposium on Mining Software Repositories, St. Louis, 17 May 2005.

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

[26] K. Crowston, H. Annabi, J. Howison, and C. Masango, "Effective work practices for Software Engineering: Free/Libre Open Source Software Development," presented at WISER Workshop on Interdisciplinary Software Engineering Research, SIGSOFT 2004/FSE-12 Conference, Newport Beach, CA, 2004.

[27] K. Crowston, H. Annabi, J. Howison, and C. Masango, "Towards a portfolio of FLOSS

project success measures," presented at Workshop on Open Source Software Engineering, 26th International Conference on Software Engineering, Edinburgh, 2004.

[28] K. Crowston, R. Heckman, H. Annabi, and C. Masango, "A structurational perspective on leadership in Free/Libre Open Source Software teams," presented at OSSCon, Genova, Italy, 2005.

[29] K. Crowston and J. Howison, "The social structure of Open Source Software development teams," presented at The IFIP 8.2 Working Group on Information Systems in Organizations Organizations and Society in Information Systems (OASIS) 2003 Workshop, Seattle, WA, 14 December 2003.

[30] K. Crowston and J. Howison, "The social structure of free and open source software development," *First Monday*, vol. 10, no. 2, 2005.

[31] K. Crowston and J. Howison, "Hierarchy and centralization in Free and Open Source Software team communications," *Knowledge, Technology & Policy*, vol. 18, no. 4, pp. 65–85, 2006.

[32] K. Crowston and J. Howison, "Hierarchy and Centralization in Free and Open Source Software team communications," *Knowledge, Technology & Policy*, In press.

[33] K. Crowston, J. Howison, and H. Annabi, "Information systems success in Free and Open Source Software development: Theory and measures," *Software Process—Improvement and Practice*, vol. 11, no. 2, pp. 123–148, 2006.

[34] K. Crowston, J. Howison, C. Masango, and U. Y. Eseryel, "Face-to-face interactions in self-organizing distributed teams," presented at Academy of Management Conference, Honolulu, HI, 2005.

[35] K. Crowston, J. Howison, C. Masango, and U. Y. Eseryel, "The role of face-to-face meetings in technology-supported self-organizing distributed teams " *IEEE Transactions on Professional Communications*, vol. 50, no. 3, 2007.

[36] K. Crowston, Q. Li, K. Wei, U. Y. Eseryel, and J. Howison, "Self-organization of teams for free/libre open source software development," *Information and Software Technology*, vol. 49, no. 6, pp. 564–575, 2007.

[37] K. Crowston and B. Scozzi, "Open source software projects as virtual organizations: Competency rallying for software development," *IEE Proceedings Software*, vol. 149, no. 1, pp. 3–17, 2002.

[38] K. Crowston and B. Scozzi, "Coordination practices for bug fixing within FLOSS development teams " presented at 1st International Workshop on Computer Supported Activity Coordination, 6th International Conference on Enterprise Information Systems, Porto, Portugal, 2004.

[39] K. Crowston and B. Scozzi, "Coordination practices within Free/Libre Open Source

Software development teams: The bug fixing process," *Journal of Database Management*, vol. Special Issue on Open Source Software, In press.

[40] K. Crowston, K. Wei, Q. Li, U. Y. Eseryel, and J. Howison, "Coordination of Free/Libre Open Source Software development," in *Proceedings of International Conference on Information Systems (ICIS 2005)*, Las Vegas, NV, USA, 2005.

[41] K. Crowston, K. Wei, Q. Li, U. Y. Eseryel, and J. Howison, "Self-organization of teams in free/libre open source software development," *Information and Software Technology Journal, Special issue on Understanding the Social Side of Software Engineering, Qualitative Software Engineering Research*, vol. 49, pp. 564–575, 2007.

[42] K. Crowston, K. Wei, Q. Li, and J. Howison, "Core and periphery in Free/Libre and Open Source software team communications," in *Proceedings of Hawai'i International Conference on System System (HICSS-39)*, Kaua'i, Hawai'i, 2006.

[43] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Communications of the ACM*, vol. 31, no. 11, pp. 1268–1287, 1988.

[44] B. Curtis, D. Walz, and J. J. Elam, "Studying the process of software design teams," in *Proceedings of the 5th International Software Process Workshop On Experience With Software Process Models*, Kennebunkport, Maine, United States, October 10-13 1990, pp. 52–53.

[45] P. S. de Souza, "Asynchronous Organizations for Multi-Algorithm Problems," Thesis, Carnegie-Mellon University 1993.

[46] C. Di Bona, S. Ockman, and M. Stone, "Open Sources: Voices from the Open Source Revolution." Sebastopol, CA: O'Reilly & Associates, 1999.

[47] A. R. Diekema, "Metadata Development and Implementation: Automatic Mapping of Educational Resources to Content Standards," presented at National Science Digital Library All Projects Meeting, Washington, DC, 15 Oct 2003.

[48] A. R. Diekema, "Evaluating Metadata from Different Perspectives," presented at Joint Conference for Digital Libraries, Workshop on Metadata Tools for Digital Resource Repositories, Chapel Hill, NC, 15 Maty 2005.

[49] A. R. Diekema, N. Balasubramanian, S. C. Harwell, and E. D. Liddy, "How to Get Standards in Your Metadata?," presented at National Science Digital Library All Projects Meeting, Denver, CO, 2005.

[50] A. R. Diekema, N. Balasubramanian, S. C. Harwell, and E. D. Liddy, "Standards Assignment and NSDL Service Integration," presented at National Science Digital Library All Projects Meeting, Denver, CO, 18 November 2005.

[51] A. R. Diekema and J. Chen, "Experimenting with the Automatic Assignment of Educational Standards to Digital Library Content," in *Proceedings of Joint Conference of*

Digital Libraries, Denver, CO, 7-11 June 2005.

[52] A. R. Diekema, S. C. Harwell, J. A. Bailey, O. Yilmazel, and E. D. Liddy, "Assigning and Aligning State and National Educational Standards," presented at National Science Digital Library Annual Meeting, Washington, DC, 19 Oct 2006.

[53] A. R. Diekema, O. Yilmazel, J. Bailey, S. C. Harwell, and E. D. Liddy, "Standards Alignment for Metadata Assignment," in *Proceedings of The Joint Conference of Digital Libraries (JCDL 2007)*, Vancouver, British Columbia, 18-23 June 2007.

[54] N. Ducheneaut, "The reproduction of Open Source Software programming communities," Thesis, University of California, Berkeley, Berkeley, CA 2003.

[55] N. Ducheneaut, "Socialization in an Open Source Software Community: A Socio-Technical Analysis," *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 4, pp. 323-368, 2005.

[56] N. Ducheneaut, "Socialization in an Open Source Software Community: a Socio-Technical Analysis," *Computer Supported Cooperative Work*, no. 14, pp. 323-368, 2005.

[57] A. H. Dutoit and B. Bruegge, "Communication Metrics for Software Development," *IEEE Transactions On Software Engineering*, vol. 24, no. 8, pp. 615–628, 1998.

[58] J. A. Espinosa, R. E. Kraut, J. F. Lerch, S. A. Slaughter, J. D. Herbsleb, and A. Mockus, "Shared mental models and coordination in large-scale, distributed software development," presented at Twenty-Second International Conference on Information Systems, New Orleans, LA, 2001.

[59] C. Ess and the AoIR ethics working committee, "Ethical decision-making and Internet research: Recommendations from the aoir ethics working committee," Association of Internet Researchers 2002.

[60] D. C. Feldman, "The Multiple Socialization of Organization Members," *The Academy of Management Review*, vol. 6, no. 2, pp. 309-318, 1981.

[61] J. Feller, "Thoughts on Studying Open Source Software Communities," in *Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective*, N. L. Russo, B. Fitzgerald, and J. I. DeGross, Eds.: Kluwer, 2001, pp. 379–388.

[62] E. Franck and C. Jungwirth, "Reconciling investors and donators: The governance structure of open source," Lehrstuhl für Unternehmensführung und -politik, Universität Zürich, Working Paper No. 8, June 2002.

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

[64] M. Grabowski and K. H. Roberts, “Risk mitigation in virtual organizations,” *Organization Science*, vol. 10, no. 6, pp. 704–721, 1999.

[65] T. L. Graves, “Inferring Change Effort from Configuration Management Databases,” 1998.

[66] R. Grishman and R. Kittredge, “Analyzing Language in Restricted Domains: Sublanguage Description and Processing.” Hillsdale, NJ: Lawrence Erlbaum, 1986.

[67] I. R. Guzman, D. Joseph, K. N. Papamichail, and J. M. Stanton, “Beliefs about IT culture: Exploring National and Gender Differences,” in *Proceedings of ACM SIGMIS Computer Personnel Research Conference*, St. Louis, MO, 19–21 Apr 2007.

[68] I. R. Guzman, J. M. Stanton, K. R. Stam, V. V., I. Yamodo, Z. Nasriah, and C. Caldera, “A Qualitative Study of the Occupational Subculture of Information Systems Employees in Organizations,” in *Proceedings of ACM Special Interest Group on Management Information Systems Computer Personnel Research Conference*, Tucson, AZ, April 2004.

[69] J. Hallen, A. Hammarqvist, F. Juhlin, and A. Chrigstrom, “Linux in the workplace,” *IEEE Software*, vol. 16, no. 1, pp. 52–57, 1999.

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

[71] I.-H. Hann, J. Roberts, and S. A. Slaughter, “Why developers participate in open source software projects: An empirical investigation,” presented at Twenty-Fifth International Conference on Information Systems, Washington, DC, 2004.

[72] A. P. Hare, *Handbook of Small Group Research*. New York: Free Press, 1976.

[73] R. Heckman, K. Crowston, U. Y. Eseryel, J. Howison, E. Allen, and Q. Li, “Emergent decision-making practices In Free/Libre Open Source Software (FLOSS) development teams,” in *Proceedings of 3rd International Conference on Open Source Software*, Limerick, Ireland, 11–15 June 2007.

[74] J. D. Herbsleb and R. E. Grinter, “Architectures, coordination, and distance: Conway's law and beyond,” *IEEE Software*, no. September/October, pp. 63–70, 1999.

[75] J. D. Herbsleb and R. E. Grinter, “Splitting the organization and integrating the code: Conway’s law revisited,” in *Proceedings of the International Conference on Software Engineering (ICSE ‘99)*, Los Angeles, CA, 1999, pp. 85–95.

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

[77] S. C. Herring, “Computer-Mediated Communication: Linguistic, Social, and Cross-

Cultural Perspectives." Philadelphia: John Benjamins, 1996.

[78] G. Hertel, S. Niedner, and S. Herrmann, "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel," University of Kiel, Kiel, Germany n.d.

[79] J. Howison, M. Conklin, and K. Crowston, "FLOSSmole: A collaborative repository for FLOSS research data and analyses," *International Journal of Information Technology and Web Engineering*, vol. 1, no. 3, pp. 17–26, 2006.

[80] J. Howison, M. S. Conklin, and K. Crowston, "OSSmole: A collaborative repository for FLOSS research data and analyses," presented at 1st International Conference on Open Source Software, Genova, Italy, 11–14 July 2005.

[81] J. Howison and K. Crowston, "The perils and pitfalls of mining SourceForge," presented at Presentation at the Workshop on Mining Software Repositories, 26th International Conference on Software Engineering, Edinburgh, Scotland, 2004.

[82] J. Howison, K. Inoue, and K. Crowston, "Social dynamics of FLOSS team communications," presented at The Second International Conference on Open Source Systems, Como, Italy, 2006.

[83] W. S. Humphrey, *Introduction to Team Software Process*: Addison-Wesley, 2000.

[84] S. L. Jarvenpaa and D. E. Leidner, "Communication and trust in global virtual teams," *Organization Science*, vol. 10, no. 6, pp. 791–815, 1999.

[85] C. F. Kemerer and S. Slaughter, "An Empirical Approach to Studying Software Evolution," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, 1999.

[86] S. Kiesler and J. Cummings, "What do we know about proximity and distance in work groups? A legacy of research," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 57–80.

[87] S. Koch and G. Schneider, "Effort, co-operation and co-ordination in an open source software project: GNOME," *Information Systems Journal*, vol. 12, no. 1, pp. 27–42, 2002.

[88] B. Kogut and A. Metiu, "Open-source software development and distributed innovation," *Oxford Review of Economic Policy*, vol. 17, no. 2, pp. 248–264, 2001.

[89] R. E. Kraut, C. Steinfield, A. P. Chan, B. Butler, and A. Hoag, "Coordination and virtualization: The role of electronic networks and personal relationships," *Organization Science*, vol. 10, no. 6, pp. 722–740, 1999.

[90] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Communications of the ACM*, vol. 38, no. 3, pp. 69–81, 1995.

[91] S. Krishnamurthy, “Cave or Community? An Empirical Examination of 100 Mature Open Source Projects,” *First Monday*, vol. 7, no. 6, 2002.

[92] G. F. Lanzara and M. Morner, “Making and sharing knowledge at electronic crossroads: the evolutionary ecology of open source,” presented at 5th European Conference on Organizational Knowledge, Learning and Capabilities, Innsbruck, Austria, 2004.

[93] G. K. Lee and R. E. Cole, “From a firm-based to a community-based model of knowledge creation: The case of Linux kernel development,” *Organization Science*, vol. 14, no. 6, pp. 633–649, 2003.

[94] E. Leibovitch, “The business case for Linux,” *IEEE Software*, vol. 16, no. 1, pp. 40–44, 1999.

[95] J. Lerner and J. Tirole, “The open source movement: Key research questions,” *European Economic Review*, vol. 45, pp. 819–826, 2001.

[96] Q. Li, K. Crowston, R. Heckman, and J. Howison, “Language and power in self-organizing distributed teams,” presented at OCIS Division, Academy of Management Conference, Atlanta, GA, 2006.

[97] E. D. Liddy, C. L. Jorgensen, E. E. Sibert, and E. S. Yu, “Sublanguage grammar in natural language processing,” in *Proceedings of RIAO '91 Conference*, Barcelona, 1991.

[98] E. D. Liddy, C. L. Jorgensen, E. E. Sibert, and E. S. Yu, “A sublanguage approach to Natural Language Processing for an expert system,” *Information Processing & Management*, vol. 29, no. 5, pp. 633–645, 1993.

[99] J. Ljungberg, “Open Source Movements as a Model for Organizing,” *European Journal of Information Systems*, vol. 9, no. 4, 2000.

[100] M. R. Louis, “Surprise and Sense Making: What Newcomers Experience in Entering Unfamiliar Organizational Settings ” *Administrative Science Quarterly*, vol. 25, no. 2, pp. 226-251, 1980.

[101] M. R. Louis, B. Z. Posner, and G. N. Powell, “The availability and helpfulness of socialization practices,” *Personnel Psychology*, vol. 36, no. 4, pp. 857-866, 1983.

[102] G. Madey, V. Freeh, and R. Tynan, “The Open Source Software development phenomenon: An analysis based on social network theory,” in *Proceedings of the Eighth Americas Conference on Information Systems* 2002, pp. 1806–1815.

[103] G. Mark, “Conventions for coordinating electronic distributed work: A longitudinal study of groupware use,” in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 259–282.

[104] M. L. Markus, B. Manville, and E. C. Agres, “What makes a virtual organization work?,” *Sloan Management Review*, vol. 42, no. 1, pp. 13–26, 2000.

[105] L. L. Martins, L. L. Gilson, and M. T. Maynard, "Virtual teams: What do we know and where do we go from here?," *Journal of Management*, vol. 30, no. 6, pp. 805-835, 2004.

[106] M. B. Miles and A. M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*, 2nd ed. Thousand Oaks: Sage Publications, 1994.

[107] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "A case study of Open Source Software development: The Apache server," in *Proceedings of the International Conference on Software Engineering (ICSE'2000)* 2000, pp. 11 pages.

[108] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies Of Open Source Software development: Apache And Mozilla," *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.

[109] G. Moody, *Rebel code—Inside Linux and the open source movement*. Cambridge, MA: Perseus Publishing, 2001.

[110] R. L. Moreland and J. M. Levine, "Socialization in Organizations and Work Groups," *Groups at Work: Theory and Research*, 2001.

[111] R. L. Moreland and J. M. Levine, "Socialization and Trust in Work Groups," *Group Processes & Intergroup Relations*, vol. 5, no. 3, pp. 185, 2002.

[112] B. A. Nejmeh, "Internet: A strategic tool for the software enterprise," *Communications of the ACM*, vol. 37, no. 11, pp. 23–27, 1994.

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

[114] S. O'Mahony and F. Ferraro, "Managing the Boundary of an 'Open' Project," in *Proceedings of Santa Fe Institute (SFI) Workshop on The Network Construction of Markets*, J. Padgett and W. Powell, Eds. 10 October 2003.

[115] M. O'Leary and J. Cummings, "The Spatial, Temporal, and Configurational Characteristics of Geographic Dispersion in Teams," presented at Academy of Management Conference, Denver, CO, 2002.

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

[117] W. J. Orlikowski, "Knowing in practice: Enacting a collective capability in distributed organizing," *Organization Science*, vol. 13, no. 3, pp. 249–273, 2002.

[118] J. Orr, *Talking About Machines: An Ethnography of a Modern Job*. Ithaca, NY: ILR Press, 1996.

[119] C. Ostroff and S. W. J. Kozlowski, "Organizational socialization as a learning process: The role of information acquisition," *Personnel Psychology*, vol. 45, no. 4, pp. 849-874, 1992.

[120] B. Pfaff, "Society and open source: Why open source software is better for society than proprietary closed source software," [Online document], 1998, Available HTTP: http://www.msu.edu/user/pfaffben/writings/anp/oss-is-better.html

[121] G. C. Prasad, "A hard look at Linux's claimed strengths…," [Online document], n.d., Available HTTP: http://www.osopinion.com/Opinions/GaneshCPrasad/GaneshCPrasad2-2.html

[122] O. Rambow, L. Shrestha, J. Chen, and C. Laurdisen, "Summarizing Email Threads.," in *Proceedings of HLT-NAACL* 2004, pp. 105–108.

[123] E. S. Raymond, "The cathedral and the bazaar," *First Monday*, vol. 3, no. 3, 1998.

[124] D. Robey, H. M. Khoo, and C. Powers, "Situated-learning in cross-functional virtual teams," *IEEE Transactions on Professional Communication*, no. Feb/Mar, pp. 51–66, 2000.

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

[126] S. Sawyer and P. J. Guinan, "Software development: Processes and performance," *IBM Systems Journal*, vol. 37, no. 4, pp. 552–568, 1998.

[127] W. Scacchi, "The software infrastructure for a distributed software factory," *Software Engineering Journal*, vol. 6, no. 5, pp. 355–369, 1991.

[128] W. Scacchi, "Understanding the requirements for developing Open Source Software systems," *IEE Proceedings Software*, vol. 149, no. 1, pp. 24–39, 2002.

[129] E. H. Schein, "Organizational Socialization and the Profession of Management," *Organizational Influence Processes*, 2003.

[130] B. Scozzi, K. Crowston, U. Y. Eseryel, and Q. Li, "Shared mental models among open source software developers," in *Proceedings of Hawai'i International Conference on System Science*, Big Island, Hawai'i, 2008.

[131] C. B. Seaman and V. R. Basili, "Communication and Organization in Software Development: An Empirical Study," Institute for Advanced Computer Studies, University of Maryland, College Park, MD, USA 1997.

[132] J. M. Stanton, I. R. Guzman, and I. Fagnot, "Internships and Occupational Commitment of College Students in IT-Related Majors," in *Proceedings of SIGMIS Computer Personnel Research Conference*, Claremont, CA, 13–15 Apr 2006.

[133] K. J. Stewart and T. Ammeter, “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*, 2002, pp. 853–857.

[134] K. J. Stewart and S. Gosain, “Impacts of ideology, trust, and communication on effectiveness in open source software development teams,” presented at Twenty-Second International Conference on Information Systems, New Orleans, LA, 2001.

[135] I. Tuomi, “Evolution of the Linux Credits File: Methodological Challenges and Reference Data for Open Source Research,” [Online document], 2002, [cited 15 November 2002], Available HTTP: http://www.jrc.es/~tuomiil/articles/EvolutionOfTheLinuxCreditsFile.pdf

[136] W. Turner, J.-P. Sansonnet, L. Gasser, and G. Ripoche, “Confidence-based organizational metrics,” presented at Workshop on Distributed Collective Practice: Building new Directions for Infrastructural Studies, CSCW 2004 2004.

[137] V. Valloppillil, “Halloween I: Open Source Software,” [Online document], 1998, Available HTTP: http://www.opensource.org/halloween/halloween1.html

[138] V. Valloppillil and J. Cohen, “Halloween II: Linux OS Competitive Analysis,” [Online document], 1998, Available HTTP: http://www.opensource.org/halloween/halloween2.html

[139] A. H. van de Ven and M. S. Poole, “Methods for studying innovation development in the Minnesota Innovations Research Program,” *Organization Science*, vol. 1, no. 3, pp. 313–335, 1990.

[140] P. C. van Fenema, “Coordination and control of globally distributed software projects,” Thesis, Erasmus University, Rotterdam, The Netherlands 2002.

[141] J. Van Maanen and E. H. Schein, “Toward a theory of organizational socialization,” *Research in Organizational Behavior*, vol. 1, pp. 209-264, 1979.

[142] P. Vixie, “Software engineering,” in *Open sources: Voices from the open source revolution*, C. Di Bona, S. Ockman, and M. Stone, Eds. San Francisco: O’Reilly, 1999.

[143] E. von Hippel, “Innovation by user communities: Learning from open-source software,” *Sloan Management Review*, no. Summer, pp. 82–86, 2001.

[144] E. von Hippel and G. von Krogh, “Exploring the Open Source Software Phenomenon: Issues for Organization Science,” Sloan School of Management, MIT, Cambridge, MA 2002.

[145] E. von Hippel and G. von Krogh, “Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science,” *Organization Science*, vol. 14, no. 2, pp. 209–213, 2003.

[146] G. von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, vol. 32, no. 7, pp. 1217--1241, 2003.

[147] D. B. Walz, J. J. Elam, and B. Curtis, "Inside a software design team: knowledge acquisition, sharing, and integration," *Communications of the ACM*, vol. 36, no. 10, pp. 63–77, 1993.

[148] M. B. Watson-Manheim, K. M. Chudoba, and K. Crowston, "Discontinuities and continuities: A new way to understand virtual work," *Information, Technology and People*, vol. 15, no. 3, pp. 191–209, 2002.

[149] P. Wayner, *Free For All*. New York: HarperCollins, 2000.

[150] E. Webb and K. E. Weick, "Unobtrusive measures in organizational theory: A reminder," *Administrative Science Quarterly*, vol. 24, no. 4, pp. 650–659, 1979.

[151] O. Yilmazel, N. Balasubramanian, S. C. Harwell, J. Bailey, A. R. Diekema, and E. D. Liddy, "Text Categorization for Aligning Educational Standards," in *Proceedings of Hawaii International Conference on System Sciences*, Waikoloa, HI, 3-6 January 2007.