

Effective work practices for Open Source Software development

We propose a study to address the general research question: what practices make some Free/Libre Open Source Software (FLOSS) development teams more effective than others? To answer this question, we propose a three-phase social science study identifying and comparing the work practices of more and less effective FLOSS development teams. The proposed research will be guided by an advisory board of FLOSS developers to ensure relevance and to help promote diffusion of our findings into practice.

As a conceptual basis for our study, we draw on Hackman's [1] model of effectiveness of work teams. Following on work by Crowston and Kammerer [2], we also use coordination theory [3] and collective mind [4] to extend Hackman's model by further elaborating team practices relevant to software development. The literature on shared mental models, collective mind theory [4] in particular, focuses our attention on actions that develop and exhibit shared understandings. Coordination theory [3] suggests identifying tasks, interdependences among tasks and resources and the coordination mechanisms that are adopted.

The proposed study has three phases. During Phase I, we will conduct a census of FLOSS projects in order to identify 1) development teams of interest and 2) concepts and relationships that seem promising for further in-depth study. During Phase II, we will conduct an in-depth multiple case study [5] on a small number of distributed FLOSS development teams (approximately 8) to test and extend the model developed in this proposal. During Phase III, we will collect and analyze data on hundreds of teams using the metrics and tools developed in Phase II, in order to generalize those findings. Phase III will also include a developer survey. Data considered will include project and developer demographics, interaction logs, code, interviews, observation and participant observation. Data will be analyzed to describe software development practices and processes, team social networks, and shared mental models of developers. Throughout the three phases we plan to check our design and preliminary results with frequent engagement with the FLOSS community through a project advisory board.

Expected intellectual contributions

The project will contribute to advancing knowledge and understanding of FLOSS development and distributed work more generally by identifying practices of effective teams. 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. Second, we use several different techniques to analyze the practices, providing different perspectives of analysis and a more reliable portrait of what happens in the development teams. Finally, we attempt to assess the relationship between the practices and the teams' effectiveness, which again represents an innovative approach in the literature on FLOSS.

Expected broader impacts

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

TABLE OF CONTENTS

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

	Total No. of Pages	Page No.* (Optional)*
Cover Sheet for Proposal to the National Science Foundation		
Project Summary (not to exceed 1 page)	1	_____
Table of Contents	1	_____
Project Description (Including Results from Prior NSF Support) (not to exceed 15 pages) (Exceed only if allowed by a specific program announcement/solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)	15	_____
References Cited	7	_____
Biographical Sketches (Not to exceed 2 pages each)	2	_____
Budget (Plus up to 3 pages of budget justification)	6	_____
Current and Pending Support	1	_____
Facilities, Equipment and Other Resources	1	_____
Special Information/Supplementary Documentation	13	_____
Appendix (List below.) (Include only if allowed by a specific program announcement/ solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)	_____	_____
Appendix Items:		

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

Effective work practices for Open Source Software development

We propose a study to address the following general research question: what practices make some Free/Libre Open Source Software (FLOSS) development teams more effective than others? To answer this question, we propose a three-phase study identifying and comparing the work practices of more and less effective FLOSS development teams, guided by social science theories of effectiveness of work teams, coordination and collective knowledge. The proposed research will be guided by an advisory board of FLOSS developers to ensure relevance and to help promote diffusion of our findings into practice.

Free/Libre Open Source Software is a broad term used to embrace software developed and released under an “open source” license allowing inspection, modification and redistribution of the software’s source without charge (“free as in beer”). Much (though not all) of this software is also “free software”, meaning that derivative works must be made available under the same unrestrictive license terms (“free as in speech”, thus “libre”). We have chosen to use the acronym FLOSS rather than the more common OSS to emphasize this dual meaning. There are thousands of FLOSS projects, spanning a wide range of applications. Due to their size, success and influence, the Linux operating system and the Apache Web Server (and related projects) are the most well known, but hundreds of others are in widespread use, including projects on Internet infrastructure (e.g., sendmail, bind), user applications (e.g., Mozilla, OpenOffice) and programming languages (e.g., Perl, Python, gcc). Many are popular (indeed, some dominate their market segment) and the code has been found to be generally of good quality [6].

Key to our interest is the fact that most FLOSS software is developed by distributed teams. Developers contribute from around the world, meet face-to-face infrequently if at all, and coordinate their activity primarily by means of computer-mediated communications (CMC) [7, 8]. These teams depend on processes that span traditional boundaries of place and ownership. The research literature on software development and on distributed work emphasizes the difficulties of distributed software development, but the case of FLOSS development presents an intriguing counter-example. What is perhaps most surprising about the FLOSS process is that it appears to eschew traditional project coordination mechanisms such as formal planning, system-level design, schedules, and defined development processes [9]. As well, many (though by no means all) programmers contribute to projects as volunteers, without working for a common organization or being paid. Characterized by a globally distributed developer force and a rapid and reliable software development process, effective FLOSS development teams somehow profit from the advantages and overcome the challenges of distributed work [10]. The “miracle of FLOSS development” poses a real puzzle and a rich setting for researchers interested in the work practices of distributed teams.

As well, FLOSS development is an important phenomena deserving of study for itself. FLOSS is an increasingly important commercial phenomenon involving all kinds of software development firms, large, small and startup. Millions of users depend on systems such as Linux and the Internet (heavily dependent on FLOSS tools), but as Scacchi [11] 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”. A 2002 EU/NSF workshop on priorities for FLOSS research identified the need both for learning “from open source modes of organization and production that could perhaps be applied to other areas” and for “a concerted effort on open source in itself, for itself” [12]. As evidenced by the attached letters of support from FLOSS developers, members of the FLOSS community are themselves interested in understanding and documenting effective practices and teams so as to improve their performance.

This proposal is organized into four sections. In the next section, we develop the problems driving our study by briefly reviewing the prior research on distributed software development and on FLOSS development in particular. We then discuss the theoretical perspectives that will guide our study, followed by the proposed research design, including details of the study phases, sampling strategy, and data elicitation and analysis plans. We conclude by presenting expected results and contributions.

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., 13], advances in information and communication technologies have been crucial enablers for recent developments of this organizational form [14]. Distributed teams seem particularly attractive for software development because the code can be shared via the systems used to support team interactions [15, 16]. While distributed teams have many potential benefits, distributed workers face many real challenges. Watson-Manheim, Chudoba, & Crowston [17] argue that distributed work is characterized by numerous discontinuities: a lack of coherence in some aspects of the work setting (e.g., organizational membership, business function, task, language or culture) that hinders members in making sense of the task and of communications from others [18], or produce unintended information filtering [19] or misunderstandings [20]. These interpretative difficulties in turn make it hard for team members to develop shared mental models of the developing project [21, 22]. A lack of common knowledge about the status, authority and competencies of team participants can be an obstacle to the development of team norms [23] and conventions [24].

The presence of discontinuities seems likely to be particularly problematic for software developers [18]. Numerous studies of the social aspects of software development teams [18, 25-28] conclude that large system development requires knowledge from many domains, which is thinly spread among different developers [25]. As a result, large projects require a high degree of knowledge integration and the coordinated efforts of multiple developers [29]. More effort is required for interaction when participants are distant and unfamiliar with each others work [30, 31]. The additional effort required for distributed work often translates into delays in software release compared to traditional face-to-face teams [32, 33]. The problems facing distributed software development teams are reflected in Conway's law, which states that the structure of a product mirrors the structure of the organization that creates it. Accordingly, splitting software development across a distributed team will make it hard to achieve an integrated product [9].

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 CMC [34]. Research has shown the importance of formal and informal coordination mechanisms and information sharing [26] 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 [35-39]. Successful distributed teams share knowledge and information and create new practices to meet the task and social needs of the members [40]. However, the processes of knowledge sharing and socialization for distributed teams are still open topics for research [e.g., 41].

Research on FLOSS development

The nascent research literature on FLOSS has addressed a variety of questions. First, researchers have examined the implications of FLOSS from economic and policy perspectives. For example, some authors have examined the implications of free software for commercial software companies or the implications of intellectual property laws for FLOSS [e.g., 42-44]. Second, various explanations have been proposed for the decision by individuals to contribute to projects without pay [e.g., 45-49]. These authors have mentioned factors such as personal interest, ideological commitment, development of skills [50] or enhancement of reputation [49]. Finally, a few authors have investigated the processes of FLOSS development [e.g., 7, 51], which is the focus of this proposal.

Raymond's [7] bazaar metaphor is the most well-known model of the FLOSS process. As with merchants in a bazaar, FLOSS developers are said to autonomously decide how and when to contribute to project development. By contrast, traditional software development is likened to the building of a cathedral, progressing slowly under the control of a master architect. While popular, the bazaar metaphor has been broadly criticized. According to its detractors, the bazaar metaphor disregards important aspects of the FLOSS process, such as the importance of project leader control, the existence of de-facto hierarchies, the danger of information overload and burnout, and the possibility of conflicts that cause a loss of interest in a project or forking [52, 53].

Recent empirical work has begun to illuminate the structure and function of FLOSS development teams. Gallivan [54] analyzes descriptions of the FLOSS process and suggests that teams rely on a variety of social control mechanisms rather than on trust. Several authors have described teams as having a hierarchical or onion-like structure [55-57], as shown in Figure 1. At the centre are the core developers, who contribute most of the code and oversee the design and evolution of the project.

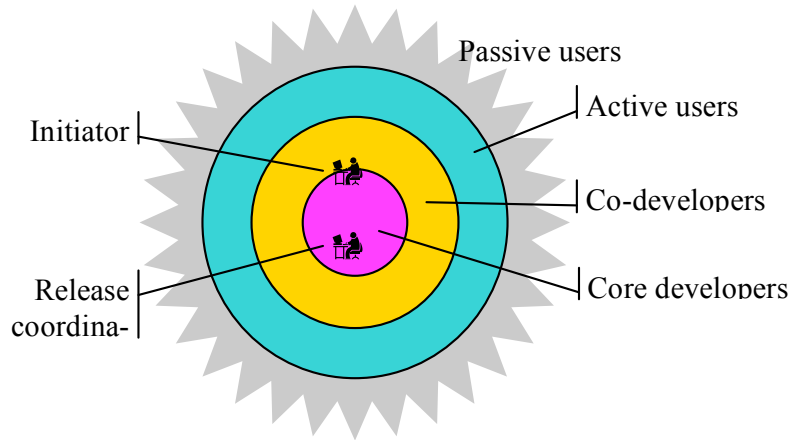


Figure 1. Hypothesized FLOSS development team structure.

The core is usually small and exhibits a high level of interaction, which would be difficult to maintain if the core group were large. Surrounding the core are the co-developers. These individuals contribute sporadically by reviewing or modifying code or by contributing bug fixes. The co-developer group can be much larger than the core, because the required level of interaction is much lower. Surrounding the developers are the active users: a subset of users who use the latest releases and contribute bug reports or feature requests (but not code). Still further from the core are the passive users. The border of the outer circle is indistinct because the nature and variety of FLOSS distribution channels makes it difficult or impossible to know the exact size of the user population. As their involvement with a project changes, individuals may move from role to role. However, core developers must have a deep understanding of the software and the development processes, which poses a significant barrier to entry [58, 59]. This barrier is particularly troubling because of the reliance of FLOSS projects on volunteer submission and “fresh blood” [60]. It is important to note that this description of a project team (Figure 1) is based on a few case studies. While the model has good face validity, it has not been extensively tested. We plan to determine how well it holds up for a wider range of projects, as part of our investigation of how the social structure of a project team influences team effectiveness.

The other major stream of research examines factors for the success of FLOSS in general (though there have been few systematic comparison across multiple projects, e.g., [61]). 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 [62-68]. In turn, the quality of the software and speed of development have been attributed to two factors: that developers are also users of the software and the availability of source code. First, FLOSS projects often originate from a personal need [69, 70], 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 [71]. (Of course, over-reliance on this mode of requirements gathering may also limit the applicability of the FLOSS model.) Second, in FLOSS projects, the source code is open to modification, enabling users to become co-developers by developing fixes or enhancements. As a result, FLOSS bugs can be fixed and features evolved quickly. Active users also play an important role [72]. Research suggests that more than 50 percent of the time and cost of non-FLOSS software projects is consumed by mundane work such as testing [73]. The FLOSS process enables hundreds of people to work on these parts of the process [74]. Intriguingly, it has been argued that the distributed nature of FLOSS development may actually lead to more robust and maintainable code. Because developers cannot consult each other easily, it may be that they make fewer assumptions about how their code will be used and thus write more robust code that is highly modularized [74].

Specific research questions

It is noteworthy that much of the literature on FLOSS has been written by developers and consultants directly involved in the FLOSS community. These contributions are significant as they point out the economic relevance of FLOSS as well as the most striking aspects of the new development process. Yet many of these studies seem to be animated by partisan spirit, hype or skepticism [75]. There are only a few well-documented case studies [e.g., 76], most of which discuss successes rather than failures. Finally, with a few exceptions [e.g., 77, 78], the proposed models are descriptive and based on a small number of cases. This is both indicative of the relative novelty of the issue and the lack of a clear theoretical framework to describe and interpret the FLOSS phenomenon [79]. Our study is intended to fill some of these gaps by providing a theoretically-based comparative study of FLOSS development practices. Based on the literature reviewed above, we have identified the following specific research questions that address our initial general question:

What practices make some distributed FLOSS development teams more effective than others? Specifically:

1. what roles and patterns of social relations,
 2. what practices for building and maintain shared mental models
 3. what practices for coordination and communication and
 4. what practices for socialization of new member
- distinguish more and less effective FLOSS project teams?

2. Conceptual development

Our study will identify work practices in effective FLOSS projects. To do so, we have chosen to analyze developers as comprising a work team. Much of the literature on FLOSS has conceptualized developers as forming communities, which is a useful perspective for understanding why developers choose to join or remain in a project. However, for the purpose of this study, we view the projects as entities that have a goal of developing a product, whose members are interdependent in terms of tasks and roles, and who have a user base to satisfy, in addition to having to attract and maintain members. These aspects of FLOSS projects suggest analyzing them as work teams. Guzzo and Dickson [80, pg. 308] defined a work team as “made up of individuals who see themselves and who are seen by others as a social entity, who are interdependent because of the tasks they perform as members of a group, who are embedded in one or more larger social system (e.g. community, or organization), and who perform tasks that affect others (such as customers or coworkers)”.

Given this perspective, we draw on Hackman’s [1] model of effectiveness of work teams as a conceptual basis for our study. This model was initially presented as sets of factors, but these factors to point to work practices that are important for team effectiveness. Following on Crowston and Kammerer [2], we also use coordination theory [3] and collective mind [4] to extend Hackman’s model by further elaborating team practices relevant to effectiveness in software development. In this section, we describe these theories, their applicability to FLOSS development and their implications for our study. We defer discussion of data elicitation and analysis to the following section.

Team effectiveness model

Researchers in social and organizational psychology have studied teams and their performance for decades and have developed a plethora of models describing and explaining team behavior and performance. One of the most widely used normative models was proposed by Hackman [1], shown in Figure 2 below. Hackman’s [1] model is broadly similar to other models [81], such as [82], [83] or [84]. However, Hackman’s model seems especially fitting because of its intended purpose of identifying factors related to team effectiveness, broadly defined, and its inclusion of team process factors.

Hackman’s [1] model is presented in an input-process-output framework. The **output** explained by the model is team effectiveness, which is clearly a key variable for our study: if we cannot distinguish more and less effective teams, we cannot identify work practices related to effectiveness. An attractive feature of this model is that effectiveness is conceptualized along multiple dimensions, not

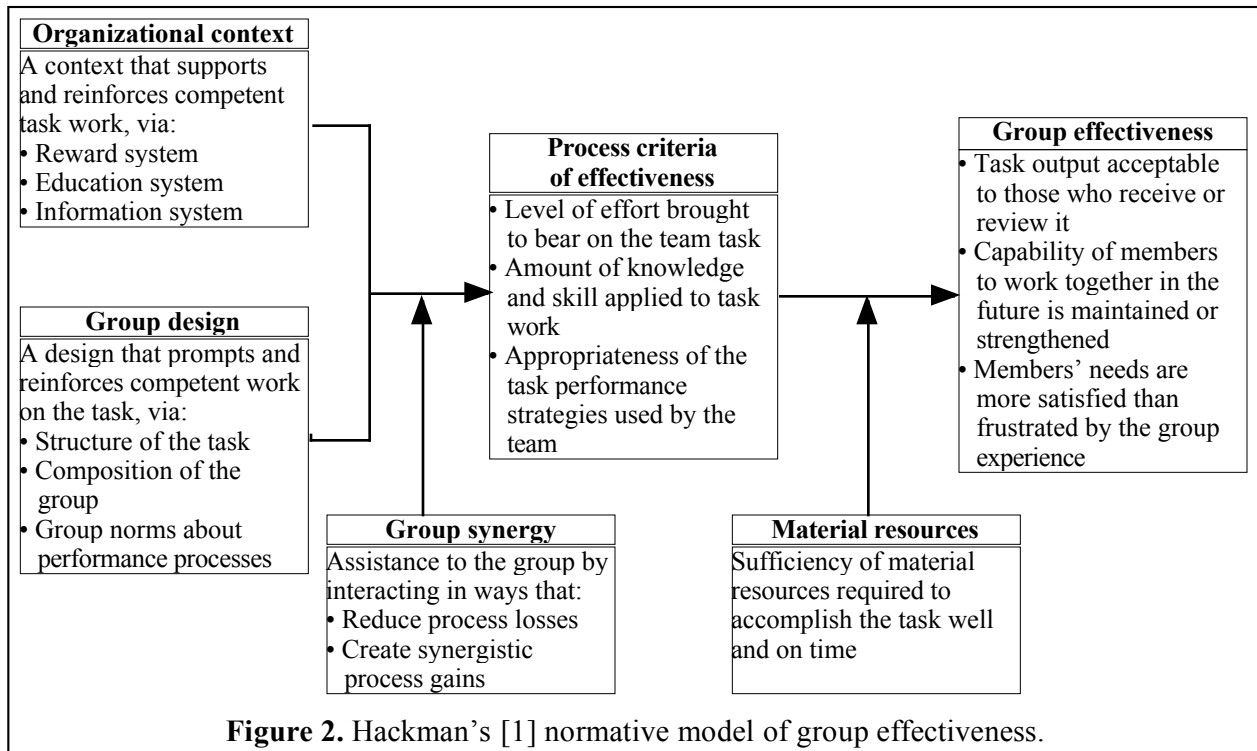


Figure 2. Hackman's [1] normative model of group effectiveness.

just task output. Hackman also includes the team's continued capability to work together and satisfaction of individual team members' personal needs. These three types of output correspond well to the effectiveness measures for FLOSS projects identified by Crowston, Annabi and Howison [85], who proposed measures including system quality (task output), developer satisfaction (satisfaction of individual needs), and number of developers, developer turnover and progress of the project through stages of development (e.g., alpha to beta to production), all indicative of the continued ability of the team to work together.

Hackman's model includes two sets of **input** factors, *organizational context* (reward, educational and information systems) and *group design* (task structure, team composition and team norms). The organizational context factors seem possibly important, though FLOSS teams typically mix members from a variety of organizational contexts, so these contextual factors may not be under the control of the projects. Therefore, we plan to focus initially on team design, which includes three promising factors to explore: task structure, team composition and team norms. All FLOSS teams perform much the same task, namely software development, but we anticipate seeing important differences in the way teams structure the task. To analyze these structures, we will use coordination theory (discussed below). As well, based on the review above, we anticipate seeing differences in practices related to team composition and development of team norms. For example, teams may differ in the roles adopted by members or in the way new members are socialized into teams.

The intermediary factors in Hackman's model are three **process criteria** (i.e., indications that the process is working as it should): "the level of effort brought to bear on the team task, amount of knowledge and skill applied to task work, and appropriateness of the task performance strategies used by the group" [1]. Prior work has noted that distributed teams often need to expend more effort to be effective, suggesting the importance of this variable. Amount of knowledge and skill applied also seem critical, though possibly difficult to measure and again perhaps not directly under the control of the project. We will again use coordination theory to analyze task performance strategies.

Finally, Hackman proposes factors that moderate the relationship between process and output, namely material resources, and between inputs and process, namely team synergy. For software development, relevant material resources would seem to be limited to development tools, which are readily available, thanks to systems like SourceForge (<http://sourceforge.net/>) and Savannah

(<http://savannah.gnu.org/>), which host thousands of projects. The review of software development presented above makes clear that practices for the development and maintenance of shared mental models will play an important role in enabling team synergy. We will apply collective mind [4] theory to conceptualize these models, as discussed below. In the remainder of this section, we will discuss the two supporting theories we will use to extend Hackman's model.

Coordination theory

As mentioned above, we will use coordination theory to analyze the structure of the tasks and coordination mechanisms used within teams. Many software process researchers have stressed the importance of coordination for software development [e.g., 25, 71]. For example, Kuwabara [86] states that, "coordination is a crucial element sustaining collective effort giving the Linux its integrity that unfolds the seemingly chaotic yet infinitely creative process of creation". The knowledge based-view of the firm [87] also emphasizes coordination mechanisms as important for integrating the knowledge of individuals into an organization's products.

Coordination theory provides a theoretical framework for analyzing coordination in processes. We will use the model presented by Malone and Crowston [3], who define coordination as "managing dependencies." They analyzed processes in terms of actors performing interdependent tasks. These tasks might also require or create resources of various types. For example, in software development, developers might require bug reports in order to create patches for the bugs. In this view, actors in organizations face coordination problems arising from interdependencies that constrain how tasks can be performed. Interdependencies can be between tasks, between tasks and the resources they need or between the resources used. Interdependencies may be inherent in the structure of the problem (e.g., components of a system may interact with each other, constraining how a particular component is designed [88]) or they may result from the assignment of tasks to actors and resources (e.g., two engineers working on the same component face constraints on the changes they can propose without interfering with each other). To overcome these coordination problems, actors must perform additional work, which Malone and Crowston [3] called coordination mechanisms, or what Faraj and Xiao [89] call coordination practices. For example, if particular expertise is necessary to fix a bug (a task-actor dependency), then a developer with that expertise must be identified and the bug routed to him or her to work on. For any given dependency, there may be a range of available mechanisms, so project teams are expected to differ in their choice of mechanisms. It is unlikely that there is a single best set of mechanisms, but rather the fit of the selected mechanisms with other team practices is expected to have implications for effectiveness.

Collective mind

The second theory we will apply is collective mind, a theory of the functioning of shared mental models. Shared mental models, as defined by Cannon-Bowers & Salas [90], "are knowledge structures held by members of a group that enable them to form accurate explanations and expectations for the task, and in turn, to coordinate their actions and adapt their behavior to demands of the task and other group members" (p. 228). Without shared mental models, individuals from different teams or backgrounds may interpret tasks differently, making collaboration and communication difficult [91] and diminishing individual contributions to the collective goal. Shared mental models are expected to lead to better team performance in general [90] and for software development in particular. Curtis, et al. [21], note that, "a fundamental problem in building large systems is the development of a common understanding of the requirements and design across the project group" (p. 52). They go on to say that, "transcripts of group meetings reveal the large amounts of time designers spend trying to develop a shared model of the design" (p. 52).

Following on work by Crowston and Kammerer [2], we intend to apply Weick and Robert's [4] collective mind theory to analyze this issue. We have chosen this theory for several reasons. First, previous conceptions of group mind have been controversial because they seemed to imply the existence of some super-individual entity [92]. By contrast, collective mind is described as an individual's "disposition to heed," hence an emphasis on "heedful" behaviors. If each member of a team has the desire and means to act in ways that further the goals and needs of the team (i.e., "heedfully"), then that team will exhibit behavior that might be described as collectively intelligent, even though it

Concepts	Specific phenomena	Source of data (see section 3)
Team effectiveness	Code quality Project usage User satisfaction Project recognition	Statistical analysis of code and project demographics
	Continued system development Group membership turnover	Statistical analysis of project demographics
	Developer satisfaction Developer recognition	Developer interviews and survey
Team design	Task structure Process activities and dependencies Actors and roles	Process mapping
	Composition of team Experience Cross-membership	Social network analysis, statistical analysis of developer demographics
	Team norms about performance Socialization of new members	Content analysis of interactions, interviews and observation
Process criteria	Number of developers Level of effort of developers (quantity and quality)	Social network analysis, statistical analysis of developer interactions and code changes
	Appropriate coordination mechanisms Team communication patterns	Process mapping, social network analysis
Team synergy	Shared mental models (representation) Socialization, narration, collaboration	Content analysis of interactions, interviews and observation

Table 1. Concepts to be explored in the study and sources of data.

is the individuals who are intelligent, not the team *per se*. Second, Weick and Roberts [4] suggest that collective mind is beneficial for situations where there is need for high reliability, non-routine work, and interactive complexity (the combination of complex interactions with a high degree of coupling), all characteristics of much of software development. Finally, the elements of the theory fit cleanly into Hackman's model, as we now discuss.

Weick and Roberts [4] identify three overlapping individual behaviours that epitomize collective mind: 1) contribution (an individual member of a team contributes to the team outcome, one of Hackman's process factors), 2) representation (individuals build personal mental models of the team and its task, which we view as an important factor for Hackman's team synergy) and 3) subordination (an individual puts the team's goals ahead of individual goals, a team norm that corresponds to Hackman's team design input). Although conceptualized separately, these three concepts overlap and reinforce one another to some degree. For example, it is difficult to imagine heedful contributions from even highly talented and motivated individuals with weak representations of the team's needs and structure. While these actions go on in any group setting, the issue for collective mind is how carefully, appropriately and intelligently they are done. To the extent they are, the team will display collective mind.

Given the importance of collective mind, we will look not only for practices that exhibit it, but also those that build and maintain it. For the later purpose, Brown and Duguid's [93] model of communities of practice seems useful. Brown and Duguid [93] suggested three overlapping social processes that underlie work practices: social construction, narration, and collaboration. Construction (or socialization) addresses the issue of people joining a team needing to understand how they fit into the process being performed (i.e., their representation, contribution and subordination). New members need to be encouraged and educated to interact with one another to develop a strong sense of "how we do things around here" (i.e., representation) [13]. Second, Brown and Duguid [93] stress

the importance of narration. To keep the collective mind strong and viable, important events must be “replayed” and reanalyzed, and the history that defines who the group is and how it does things (representation) must be continually reinforced, reinterpreted, and updated and shared with newcomer. Finally, Brown and Duguid [93] stress the importance of collaboration, based on narration, thus leading to the team synergy identified in Hackman’s model.

Table 1 summarizes the constructs we will explore in the study. In the following section we will discuss how we plan to elicit and analyze data to address these constructs, as indicated in the final column of Table 1.

3. Research design

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

Software development is a vast topic. As an initial focus for our work, we plan to examine the software bug fixing process, which provides “a microcosm of coordination problems” [94]. The bug fixing process has been chosen as a starting point because a quick response to bugs has been mentioned as a particular strength of the FLOSS process: as Raymond [7] puts it, “given enough eyeballs, all bugs are shallow”. As well, it is a process that involves the entire community, the core and co-developers as well as active users. Finally, it is one that the PI [94] and others [e.g., 95] have previously studied in a commercial environment, providing a baseline for comparisons. However, as the project progresses, we plan to expand our examination to include practices related to the entire software development process.

Overall design

To address our research questions, we will conduct a three-phase study of FLOSS project teams. During Phase I, we will conduct a census of FLOSS projects in order to identify 1) development teams of interest and 2) concepts and relationships that seem promising for further in-depth study. During Phase II, we will conduct an in-depth multiple case study [5] on a small number of distributed FLOSS development teams (approximately 8) to test and extend the model developed above. During Phase III, we intend to collect and analyze data on hundreds of teams using the metrics and tools developed in Phase II in order to generalize those findings. Phase III will also include a developer survey. Throughout the three phases we plan to check our design and preliminary results with frequent engagement with the FLOSS community through a project advisory board of developers.

Phase I: Searching for interesting teams and variables

In Phase I, we will examine a large number of projects to identify those that will be most interesting for further study, and to identify specific practices that seem to be most relevant for further understanding effectiveness. In order to provide a reliable basis for generalization, the sample in this phase should be as large and as representative as possible. Ideally, we would analyze the population of FLOSS projects, which unfortunately is impossible to identify. Instead, we will use SourceForge and Savannah as primary sources, along with a number of projects that use their own tools, e.g., the Apache Foundation projects. Because many SourceForge projects are effectively defunct, we will restrict our attention in this phase to active projects.

Automatic data collection will be needed to gather data from such a large sample. We will develop scripts to collect and extract data automatically. Fortunately, a large volume of raw data is easily obtained in machine-readable format, though the requirement for automatic extraction will limit the kinds of data that we will be able to elicit during this phase. As other researchers are also examining the problem of automatic data collection, we intend to work with them rather than building all tools from scratch. Similarly, we plan to make our tools and raw data available to other researchers under a FLOSS license. We also plan to explore the possibility of working with Syracuse University’s Center for Natural Language Processing to develop tools for automatically analyzing unstructured textual data contained in emails, bug reports and other textual data [e.g., 96].

The output of Phase I will be an indication of which variables seem to be good measures of project effectiveness and which other variables seem to be related, suggesting relevant practices for further study. As an example of this approach, we have conducted a pilot study using Social Network Analysis (SNA) to analyze the structure of FLOSS development teams from their interactions in Bug Trackers and developer mailing lists [97]. For the study, we downloaded data from all SourceForge projects that had more than 7 listed developers and more than 100 bugs in their bug tracking system (these two filters eliminated all but 140 projects). The analysis revealed that teams vary significantly in centralization, with some teams highly centralized (all communications revolving around just one or two developers) and some relatively decentralized (with interactions between project members without a clear centre, or with multiple centers). Furthermore we found that centralization was negatively correlated with project size. This pilot is indicative of the analysis intended for Phase I, as we identified and measured a theoretically interesting concept (team centralization) and found interesting patterns. This analysis alone does not provide much insight into how the underlying practices work, but instead suggests phenomena that merit deeper investigation in Phase II. For example, the negative correlation between size and centralization suggests that teams experience episodes of growth and modularization (i.e., they are shallots rather than the onion of Figure 1), but more intensive investigation is needed to understand these processes and their implications for effectiveness.

In addition, in Phase I we will be able to identify a small number of projects that seem to have an interesting combination of factors, e.g., projects that seem to be particularly effective or that have interesting combinations of factors for comparison. For example, the pilot study identified a few teams that are especially centralized and especially decentralized for the number of contributors.

Phase II: Theory refinement and development through comparative case studies

The goal of Phase II is to examine the underlying practices of a smaller number of projects in more detail. In this phase we will employ a theoretical sampling strategy to choose a few FLOSS development teams to study in depth. Studying 8 teams would allow us to compare teams with combinations of three factors identified in Phase I (a 2x2x2 comparison). By limiting the number of projects, we will be able to use more labour-intensive data analysis approaches. Though the specific projects to be selected will be identified in Phase I, we can suggest some considerations that seem likely to be important.

- First, to be able to address our research question, we will compare more and less effective teams based on the measures of effectiveness proposed by Crowston et al. [85].
- Second, we will focus on teams with multiple core developers in which coordination and socialization problems have become pressing. A preliminary examination of SourceForge projects [63] suggests 8 or more core developers as an initial cutoff.
- Third, we will choose projects for which the data we need for our analysis are available (some projects do not allow access to mailing list archives and other data). We plan to check with developers to ensure that these archives give a fair picture of the interactions. Since some projects do have face-to-face meetings, our ability to attend these meetings and observe the interactions will also be a factor.
- Finally, since we plan to interview developers, the willingness of developers to participate in the study will be important. Fortunately, we have already secured letters of support from a number of developers.

This phase will be a combination of theory testing (following up on ideas from the literature reviewed above) and theory development (examining the data inductively to develop new ideas). The analysis will be aimed at the identification and detailed description of work practices that affect team effectiveness, and development of a set of metrics and measurement tools for these practices.

Phase III: Developer and project study

In the final phase, we will examine a larger number of projects to test the generality of the practices identified in Phase II and their relationship to project effectiveness. We will also follow up on the developer interviews through a survey to test hypotheses formulated during Phase II.

Project study. In this phase, we plan to examine a large number of projects for evidence of the practices identified in Phase II. Again, we would like to have as large a sample as possible. However, if important data have to be hand coded, it will be necessary to restrict the sample size for tractability. In this case, we plan to extend the sampling strategy of Phase II and examine more and less effective projects with variation along relevant dimensions identified in Phases I and II. In order to provide a reliable basis for generalization, attention will be paid to developing a representative sample of projects, which requires developing a sampling frame, stratification strategies, etc. We plan to analyze at least 125 projects¹ to provide sufficient statistical power for any comparisons we might make, though the precise sample size will depend on the coding effort required per project, which is difficult to estimate in advance.

Developer study. The second component of Phase III will be a survey of FLOSS developers to test hypotheses developed in Phase II. Because of the difficulty of creating a sampling frame of FLOSS developers, we will likely have to rely on a convenience sample (e.g., by soliciting respondents from FLOSS mailing lists and websites, at conferences, etc.). Although we will use a convenience sample, we will be able to assess its representativeness by comparing characteristics of our respondents to other published reports [e.g., 98]. A sample of about 1000 will be required to estimate proportions with a confidence interval of $\pm 3\%$. To encourage a large response, we plan to seek the support of FLOSS thought leaders (e.g., from our advisory board or through SlashDot or SourceForge) to promote participation. Again, there are several other research teams interested surveying FLOSS developers, so we plan to collaborate rather than compete for attention (especially because the attention of developers may be quite limited). For ease of administration, the questionnaire will be carried out via the Web. Fortunately, all members of the target population can be assumed to have Internet access and to be comfortable with the use of Web, so this choice of administration should not create any sampling biases.

Data collection

Practices are often hard to study because they are taken for granted, and so escape intense observation. They go on all around us, but without notice unless something goes wrong. For on-line teams though, observation is facilitated because much of the team's interactions are funneled through a CMC system, and so structured and captured, as are the results of their work. Retrospective comparisons can be easily made by comparing data captured at different times, unbiased by the possibly selective recollections of informants. Our problem then is ensuring that these interactions present a complete picture of the team and then making sense of the vast pool of data created in the course of developers' interactions to answer interesting questions about their practices. To explore the concepts identified in the conceptual development section of this proposal, we will collect a wide range of data: project demographics, developer demographic data, interaction logs, code, project plans and procedures, as well as developer interviews, observation and participant observation. In the remainder of this section, we will briefly review each source. Table 2 shows the mapping from each data source to analysis. The table also indicates which data sources will be used in the three phases of the study.

Project demographics. We will collect basic descriptive data about each project, such as its topic, intended use environment, programming language, etc. Often these data are self-reported by the developers to guide potential users (e.g., on SourceForge or FreshMeat, <http://freshmeat.net/>); in other cases, they can be inferred. We will also collect data indicative of the success of the project [61], such as its level of activity, number of downloads and development status, as well as any user ratings, such as FreshMeat user ratings. Again, SourceForge explicitly tracks these figures, but for other projects they may have to be inferred.

¹ A sample size of 63 per group is necessary to have an 80% chance of detecting an effect size (i.e., 80 % power) of 0.5 standard deviations between two groups at a 5% confidence level (assuming equal variance). The same sample size at the same power and confidence levels will detect a correlation of 0.22 or an effect size of 0.25 standard deviations per one standard deviation change in the independent variable in a regression study (from <http://calculators.stat.ucla.edu/powercalc/> and http://hedwig.mgh.harvard.edu/sample_size/size.html).

Developer demographic data. We will collect the list of developers for each project and their assigned roles, if any, plus any demographic information available. SourceForge collects skills ratings for a few developers; since only a minority of developers are rated at all, these are mostly interesting as a reflection of how well known a developer is. We also will collect developer's PGP or GnuPG key to examine the web of trust as a reflection of the developer's social network [99] (see <http://www.chaosreigns.com/code/sig2dot/> for examples).

Developer interactions logs. The most voluminous source of data will be collected from archives of CMC tools used to support the team's interactions for FLOSS development work [33, 74]. These data are useful because they are unobtrusive measures of the team's behaviours [100]. Mailing list archives will be examined, as email is a primary tool used to support team coordination [101]. Such archives contain a huge amount of information: e.g., the Linux kernel list receives 5-7000 messages per month. From mailing lists, we will extract the date, sender and any individual recipient's names, the sender of the original message, in the case of a response, and text of each message. From bug tracking systems (e.g., Apache's GNATS, Linux kernel's Jitterbug, Mozilla's Bugzilla as well as Sourceforge's Tracker) we will extract data about bug typologies, who submitted bugs, who fixed them and the steps in the bug fixing process. We will examine features request archives and logs from other interaction tools, such as chat sessions. While in most cases these archives are public, we plan to consult with the Syracuse University Human Subjects Institutional Review Board to determine what kind of consent should be sought before proceeding with analysis.

Source code. A major advantage of studying open source software is that we have access to the source code itself. Many 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 kinds of contributions to understand the software structure and the date and name of the contributors to understand the role of individual developers [76, 102, 103]. Raw code poses numerous challenges to interpretation [104]. For example, not all projects assign authorship in the CVS tree. Again, we intend to leverage our analysis with work being carried out by other researchers [e.g., 105].

Project plans and procedures. Many projects have stated release plans and proposed changes. Such data are often available on the project's documentation web page or in a "status" file used to keep track of the agenda and working plans [60]. For example, Scacchi [11] examined requirements documentation for FLOSS projects. We will also examine any explicitly stated norms, procedures or rules for taking part in a project, such as the process to submit and handle bugs, patches or feature request. Such procedures are often reported on the project's web page (e.g., <http://dev.apache.org/guidelines.html>).

Developer attitudes and opinions. While the data sources listed above will provide an extensive pool of data, they are all indirect. Interviews and surveys are important to get rich, first-hand data about developers' perceptions and interpretations. We plan to conduct interviews with key informants in the selected projects. Interviews will be conducted mainly by e-mail, but we also plan to attend one or two FLOSS conferences each year (e.g., the *O'Reilly Open Source Convention* or *ApacheCon*) to interview FLOSS developers face-to-face. The interviews will be scheduled after the initial round of data analysis to ensure that we have a sufficient understanding of the process to be able to pose intelligent questions. As part of the interviews protocol, we will employ the critical incident technique, in which developers are asked to describe personally experienced specific incidents which had an important effect on the final outcome [106]. We will also explore the developer's initial experiences of participation in FLOSS, the social structure and norms of the team, processes of knowledge exchange and socialization (especially the role of observation, which leaves no traces in the interaction logs), knowledge of other members' participation [107, 108] and impressions of project effectiveness. As well, interviews will be used to verify that the archives of interaction data give a fair and reasonably complete record of day-to-day interactions. In later phases of the project, a Web survey will be used to elicit attitudes and opinions from a large sample of developers.

Observation. We have found from our initial pilot study (described below under Results from Prior Funding) that developers interact extensively at conferences. Indeed, Nardi and Whittaker

Data source	Analysis approach	Phase		
		I	II	III
Project demographics	Statistical	✓		✓
Developer demographics	Statistical	✓		✓
Developer interaction logs	Social network analysis	✓		✓
	Content analysis, process mapping		✓	✓
Source code	Process mapping	✓	✓	✓
	Code quality			
Project plans and procedures	Content analysis		✓	✓
Developer interviews	Content analysis, process mapping, cognitive mapping		✓	
Developer survey	Statistical			✓
Observation of developer interactions	Content analysis, process mapping, cognitive mapping		✓	
Participant observation	Content analysis, process mapping, cognitive mapping		✓	

Table 2. Data elicitation and analysis by project phase.

[109] note the importance of face-to-face interactions for sustaining social relations in distributed teams. The FreeBSD developer Poul-Henning Kamp has also stated that phone calls can be occasionally used to solve complex problems [110]. These interactions are a small fraction of the total, but they may still be crucial to understanding the team’s practices. We plan to use attendance at developer conferences as an opportunity to observe and document the role of face-to-face interaction for FLOSS teams.

We also intend to carry out a virtual ethnographic study of developer socialization and interaction. One student involved with the project has already virtually joined several development teams (with the permission of the project leaders and the knowledge of other members) and is currently participating in their normal activities while observing and recording these activities (following a protocol approved by the Syracuse University Human Subjects Review Board). In this way, we will study and learn first hand the socialization and coordination practices of these teams. We will track these teams through the various stages of development status, from planning through production/stable stage, observing how new members join the teams and how they contribute to the team output.

Analysis

While voluminous, the data described above are mostly at a low level of abstraction. The collected data will be analyzed using a variety of techniques in order to raise the level of conceptualization to fit the theoretical perspectives described in section 2 and to address our research questions.

Statistical analysis. Quantitative data will be analyzed statistically. For example, based on the collected data, we will develop a scale for project effectiveness (e.g., combining the number of users or downloads, level of activity, development status, users’ ratings and developer ratings), so as to distinguish between more and less effective projects. This measure is a key variable in our study, so we plan to spend some time exploring the implications of various alternative measures (as well as augmenting quantitative data with qualitative data). For example, we have started an analysis of time required to fix bugs as one possible indication of the effectiveness of the development processes. Fortunately, for the purpose of Phase I we need only to identify more or less effective teams rather than make fine distinctions in the precise level of effectiveness, so the measure need not be extremely reliable right away. Variation among projects will be examined statistically to identify practices associated with effective teams.

Content analysis. In Phase II, the project will rely heavily on content analysis of the text in interaction archives and interviews to develop insights on the extent and development of common knowledge, coordination and communication practices and socialization (e.g., the way projects are created, introduction of new members, members leaving and community building). Data will be analyzed following the process suggested by Miles and Huberman [111], iterating between data collection, data reduction (coding), data display, and drawing and verifying conclusions. The researchers will develop an initial content analytic framework to discover the patterns of the various variables (suggested in phase I) present in the data. The initial (deductive) framework will be based on work on Asynchronous Learning Networks investigating social, cognitive and instructional processes of virtual teams [112]. In addition we will incorporate indicators from content analytic frameworks previously used to investigate shared mental models [e.g., 113]. We will start the data analysis using the initial content analytic scheme and modify the scheme as new categories and indicators emerge in the data [111]. Further categories will be added and other data will be collected as preliminary findings in the analysis suggest. We will use the thematic unit of analysis while conducting the content analysis to capture the various elements of the variables under investigation as appropriate. To increase the validity and reliability of the coding scheme we will conduct intercoder reliability tests and modify the content analytic scheme until we reach an 85% agreement level [114].

Social network analysis (SNA). SNA will be used to analyze patterns of interactions (e.g., who responds to whose email) in order to reveal the structure of the social network of projects. Madey, Freeh & Tynan [115] applied this technique to connections between projects, but not within projects. A pilot study using this technique was describe above. We are particularly interested in using social network information to identify various structural roles in the team and how individuals fill these roles over time. This analysis of structural roles should provide a useful counterpoint to descriptions of formal roles and process roles described above. We will assess an individual's centrality and the project's hierarchy, which seems to mediate the effect of role and status on individual performance within virtual teams [14], the way contributions are distributed among developers and the roles assumed by core developers. Analysis of these aspects is important to assess the general applicability of studies such as Mockus et al. [76], who argue that the development community participation in the Apache project is more significant in defect repairing than in the development of new functionalities (p. 322). The results of such analyses will support us in the identification of the social relations patterns and the way such patterns develop.

Process maps. The open source software development *processes* will be mapped based on an inductive coding of the steps involved. For example, to map the bug fixing process, we will examine how various bugs were fixed as recorded in the bug logs, email messages and the code. Van de Ven and Poole [116] describe in detail the methods they used to develop and test a process theory of how innovations develop over time. Yamauchi et al. [117] coded messages to understand the development processes of two FLOSS projects. Process traces can be clustered using optimal matching procedures [118] to develop clusters of processes. These process descriptions can be enriched with descriptions of the process from developers' reports of critical incidents and of the process in general [119].

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

Cognitive maps. *Cognitive maps* will be developed from interview data to represent and compare the mental models of the developers about the project and project team so as to gauge the degree of

common knowledge and the development of shared mental models [120-123]. Metrics (e.g., number of heads, tails, domain and centrality) provided by existing software packages (e.g. Decision Explorer or CMAP2) and ad hoc developed metrics will be used to analyze and compare the different maps. In particular, the comparisons among different team members' maps will provide insights about eventual shared mental models and collective mind acting within teams. We will also derive collective maps for each project. Collective maps usually represent perspectives that are common to all the members of a team. Shared perspectives derive from the comprehension of mutual positions and roles, which are fundamental to create synergies within the team. Collective maps for more and less effective team will be compared so as to explore the relationship between the existence of collective minds and project effectiveness. The PI has some experience studying mental models [2] but for this analysis in particular will work with a collaborator, Professor Barbara Scozzi, as discussed below.

Work plan

Based on preliminary assessment of the effort required, we are requesting funding for three graduate students and two undergraduate students. The two undergraduate students will be employed for 10 hours/week each during the 30 weeks of the academic year, for a total of about 600 hours (900 hours in two years). The two undergraduate students will work on development of Perl scripts to download and process data, on data management and on the survey Website. The graduate students will devote 50% effort during the academic year and 100% effort during the summers, for a total of 3300 hours/year (6600 hours in two years). Two of the graduate students will support the principal investigator in sample selection, definition of constructs and variables, and will have primary responsibility for data collection and analysis, under the oversight of the PI. The third graduate student will be assigned to carry out a virtual ethnographic study of project teams. The principal investigator will work one-third-time on the project during the summers, 1.0 months per year. Summers will be devoted to sample selection, interviews and publication of results. The PI will devote 10% of effort during the academic year to project management and oversight (1/2 day / week, supported by Syracuse University). A timeline of the project, including a break down of the work in to subphases, is included as part of the budget justification.

These activities, in particular those related to the analysis of coordination practices and cognitive models within the FLOSS development teams, will be carried out with the assistance of an international collaborator, Dr. Barbara Scozzi of the Department of Mechanical and Business Engineering, Polytechnic of Bari, Italy (please see the supporting documents section for a letter of support and vitae; no funding is being requested from NSF to support Dr. Scozzi). Dr. Scozzi has collaborated with the PI on a study of FLOSS project success factors [63] and her competencies in the analysis of coordination practices within business processes [124, 125] and cognitive mapping [126, 127] will be particularly valuable for this project.

4. Conclusion

In this proposal, we develop a conceptual framework and a research plan to investigate work practices within distributed FLOSS development teams. We are particularly interested in coordination practices, social networks, and processes through which shared mental models are developed within the teams, as these are considered as important success factors for software development.

Expected intellectual merits

The project will contribute to advancing knowledge and understanding of distributed teams by identifying practices of effective FLOSS teams. The study has three main strengths. First, we fill 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. Second, we use several different techniques to analyze the practices, providing different perspectives of analysis and a more reliable portrait of what happens in the development teams. Moreover, some of data analysis techniques, such as cognitive maps, critical incident theory and social network theory have not yet been used with FLOSS teams. Finally, we will attempt to assess the relationship between the practices and the teams' effectiveness, which again represents an innovative approach in the literature on FLOSS.

Expected broader impacts

The project has numerous broader impacts. The project will benefit society by identifying effective practices for FLOSS development, an increasingly important approach to software development. The study will also shed light on effective practices for distributed work teams in general, which will be valuable for managers who intend to implement such an organizational form. The effective practices can serve as guidelines (in team governance, task coordination, communication practices, mentoring, etc.) to improve performance and foster innovation. Understanding these questions is important because a digital society entails an increased use of distributed teams for a wide range of knowledge work. Distributed work teams potentially provide several benefits but the separation between members of distributed teams creates difficulties in coordination, collaboration and learning, which may ultimately result in a failure of the team to be effective [37, 38, 128, 129]. For the potential of distributed teams to be fully realized, research is needed on the practices of effective teams. 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 how collaborations are managed in 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 [130] note, in these teams, “developers collectively acquire and develop new skills and experiences”. To ensure that our study has a significant impact, we plan to broadly disseminate results through journal publications, conferences, workshops and on our Web pages. These results could also potentially be incorporated into the curricula of the professional masters degrees of the Syracuse University School of Information Studies, which are taught on-line and thus involve distributed teams. In order to improve infrastructure for research, we also plan to make our tools and raw data available to other researchers. The project will promote teaching, training, and learning by including graduate and undergraduate students in the research project. These students will have the opportunity to develop skills in data collection and analysis.

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

Results from prior NSF funding

Kevin Crowston has been funded by three NSF grants within the past 48 months. The most recent is IIS-0341475, *SGER: Effective work practices for Open Source software development* (\$12,052, 1 September 2003 to 31 August 2004). This small grant has provided support for travel to conferences (e.g., *ApacheCon*) to observe, interview and seek support from developers and to present preliminary results, and for the purchase of data analysis software, supporting the initial results reported in this proposal. This work has resulted in an accepted conference paper [85], with additional papers in preparation [e.g., 97].

Earlier support came from IIS-9732799 (\$69,997, September 1, 1998 to February 29, 2000) and IIS-0000178 (\$269,967, July 1, 2000 to June 30, 2003), both entitled *Towards Friction-Free Work: A Multi-Method Study of the Use of Information Technology in the Real Estate Industry*. The goal of that study was to examine how the pervasive use of information and communication technologies (ICT) in the real-estate industry changes the way people and organizations in that industry work. Initial fieldwork resulted in several journal articles [131-133] and numerous conference presentations [e.g., 134, 135]. We are now analyzing the results of a survey administered in spring 2003.

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

References cited

- [1] J. R. Hackman, "The design of work teams," in *The Handbook of Organizational Behavior*, J. W. Lorsch, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1986, pp. 315–342.
- [2] K. Crowston and E. Kammerer, "Coordination and collective mind in software requirements development," *IBM Systems Journal*, vol. 37, pp. 227–245, 1998.
- [3] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *Computing Surveys*, vol. 26, pp. 87–119, 1994.
- [4] K. E. Weick and K. Roberts, "Collective mind in organizations: Heedful interrelating on flight decks," *Administrative Science Quarterly*, vol. 38, pp. 357–381, 1993.
- [5] K. M. Eisenhardt, "Building theory from case study research," *Academy of Management Review*, vol. 14, pp. 532–550, 1989.
- [6] I. Stamelos, L. Angelis, A. Oikonomou, and G. L. Bleris, "Code quality analysis in open source software development," *Information Systems Journal*, vol. 12, pp. 43–60, 2002.
- [7] E. S. Raymond, "The cathedral and the bazaar," *First Monday*, vol. 3, 1998.
- [8] P. Wayner, *Free For All*. New York: HarperCollins, 2000.
- [9] 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: ACM, 1999, pp. 85–95.
- [10] 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), 1998.
- [11] W. Scacchi, "Understanding the Requirements for Developing Open Source Software Systems," *IEE Proceedings Software*, vol. 149, pp. 24–39, 2002.
- [12] R. A. Ghosh, "Free/Libre and Open Source Software: Survey and Study. Report of the FLOSS Workshop on Advancing the Research Agenda on Free / Open Source Software," European Commission, <http://www.infonomics.nl/FLOSS/report/workshopreport.htm>, 2002.
- [13] 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.
- [14] M. K. Ahuja, K. Carley, and D. F. Galletta, "Individual performance in distributed design groups: An empirical study," presented at SIGCPR Conference, San Francisco, 1997.
- [15] B. A. Nejme, "Internet: A strategic tool for the software enterprise," *Communications of the ACM*, vol. 37, pp. 23–27, 1994.
- [16] W. Scacchi, "The Software Infrastructure for a Distributed Software Factory," *Software Engineering Journal*, vol. 6, pp. 355–369, 1991.
- [17] 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, pp. 191–209, 2002.
- [18] P. C. van Fenema, "Coordination and control of globally distributed software projects," Erasmus University, Rotterdam, The Netherlands, Doctoral Dissertation 2002.
- [19] P. S. de Souza, "Asynchronous Organizations for Multi-Algorithm Problems," Department of Electrical and Computer Engineering, Carnegie-Mellon University, Doctoral Thesis 1993.
- [20] 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.

- [21] B. Curtis, D. Walz, and J. J. Elam, "Studying the Process of Software Design Teams," in *Proceedings*, 1990, pp. 52–53.
- [22] 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.
- [23] D. Bandow, "Geographically Distributed Work Groups and IT: A Case Study of Working Relationships and IS Professionals," in *Proceedings of the SIGCPR Conference*, 1997, pp. 87–92.
- [24] 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.
- [25] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *CACM*, vol. 31, pp. 1268–1287, 1988.
- [26] 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, pp. 63–77, 1993.
- [27] W. S. Humphrey, *Introduction to team software process*: Addison-Wesley, 2000.
- [28] S. Sawyer and P. J. Guinan, "Software development: Processes and performance," *IBM Systems Journal*, vol. 37, pp. 552–568, 1998.
- [29] F. P. Brooks, Jr., *The Mythical Man-month: Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1975.
- [30] 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.
- [31] 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*, 2000, pp. 10 pages.
- [32] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "A Case Study of Open Source Software Development: The Apache Server," in *Proceedings of ICSE'2000*, 2000, pp. 11 pages.
- [33] 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.
- [34] 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., In press.
- [35] M. Grabowski and K. H. Roberts, "Risk mitigation in virtual organizations," *Organization Science*, vol. 10, pp. 704–721, 1999.
- [36] J. D. Herbsleb and R. E. Grinter, "Architectures, coordination, and distance: Conway's law and beyond," *IEEE Software*, pp. 63–70, 1999.
- [37] S. L. Jarvenpaa and D. E. Leidner, "Communication and trust in global virtual teams," *Organization Science*, vol. 10, pp. 791–815, 1999.
- [38] 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, pp. 722–740, 1999.
- [39] 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.

- [40] D. Robey, H. M. Khoo, and C. Powers, "Situated-learning in cross-functional virtual teams," *IEEE Transactions on Professional Communication*, pp. 51–66, 2000.
- [41] W. J. Orlikowski, "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing," *Organization Science*, vol. 13, pp. 249–273, 2002.
- [42] C. Di Bona, S. Ockman, and M. Stone, "Open Sources: Voices from the Open Source Revolution." Sebastopol, CA: O'Reilly & Associates, 1999.
- [43] B. Kogut and A. Metiu, "Open-source software development and distributed innovation," *Oxford Review of Economic Policy*, vol. 17, pp. 248–264, 2001.
- [44] J. Lerner and J. Tirole, "The open source movement: Key research questions," *European Economic Review*, vol. 45, pp. 819–826, 2001.
- [45] 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.
- [46] 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.
- [47] J. Bessen, "Open Source Software: Free Provision of Complex Public Goods," Research on Innovation July 2002.
- [48] 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.
- [49] M. L. Markus, B. Manville, and E. C. Agres, "What makes a virtual organization work?," *Sloan Management Review*, vol. 42, pp. 13–26, 2000.
- [50] J. Ljungberg, "Open Source Movements as a Model for Organizing," *European Journal of Information Systems*, vol. 9, 2000.
- [51] 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.
- [52] N. Bezroukov, "Open source software development as a special type of academic research (critique of vulgar raymondism)," *First Monday*, vol. 4, 1999.
- [53] N. Bezroukov, "A second look at the Cathedral and the Bazaar," *First Monday*, vol. 4, 1999.
- [54] M. J. Gallivan, "Striking a balance between trust and control in a virtual organization: A content analysis of open source software case studies," *Information Systems Journal*, vol. 11, pp. 277–304, 2001.
- [55] J. Y. Moon and L. Sproull, "Essence of distributed work: The case of Linux kernel," *First Monday*, vol. 5, 2000.
- [56] A. Cox, "Cathedrals, Bazaars and the Town Council," <http://sladhot.org/features/98/10/13/1423253.shtml>, 1998.
- [57] C. Gacek, T. Lawrie, and B. Arief, "The many meanings of Open Source," Centre for Software Reliability, Department of Computing Science, University of Newcastle, Newcastle upon Tyne, United Kingdom, Unpublished manuscript n.d.
- [58] R. T. Fielding, "The Apache Group: A case study of Internet collaboration and virtual communities," <http://www.ics.uci.edu/fielding/talks/ssapache/overview.htm>, 1997.
- [59] F. Hecker, "Mozilla at one: A look back and ahead," <http://www.mozilla.org/mozilla-at-one.html>, 1999.
- [60] D. Cubranic and K. S. Booth, "Coordinating Open-Source Software Development," presented at Proceedings of the 7th IEEE Workshop on Enabling Technologies: Infrastructure

- for Collaborative Enterprises, 1999.
- [61] 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.
 - [62] V. Valloppillil, "Halloween I: Open Source Software," <http://www.opensource.org/halloween/halloween1.html>, 1998.
 - [63] K. Crowston and B. Scozzi, "Open source software projects as virtual organizations: Competency rallying for software development," *IEE Proceedings Software*, vol. 149, pp. 3–17, 2002.
 - [64] G. C. Prasad, "A hard look at Linux's claimed strengths...", <http://www.osopinion.com/Opinions/GaneshCPrasad/GaneshCPrasad2-2.html>, n.d.
 - [65] V. Valloppillil and J. Cohen, "Halloween II: Linux OS Competitive Analysis," <http://www.opensource.org/halloween/halloween2.html>, 1998.
 - [66] J. Hallen, A. Hammarqvist, F. Juhlin, and A. Chrigstrom, "Linux in the workplace," *IEEE Software*, vol. 16, pp. 52–57, 1999.
 - [67] E. Leibovitch, "The business case for Linux," *IEEE Software*, vol. 16, pp. 40–44, 1999.
 - [68] B. Pfaff, "Society and open source: Why open source software is better for society than proprietary closed source software," <http://www.msu.edu/user/pfaffben/writings/anp/oss-is-better.html>, 1998.
 - [69] G. Moody, *Rebel code—Inside Linux and the open source movement*. Cambridge, MA: Perseus Publishing, 2001.
 - [70] 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.
 - [71] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Communications of the ACM*, vol. 38, pp. 69–81, 1995.
 - [72] T. O'Reilly, "Lessons from open source software development," *Communications of the ACM*, vol. 42, pp. 33–37, 1999.
 - [73] T. Shepard, M. Lamb, and D. Kelly, "More testing should be taught," *Communication of the ACM*, vol. 44, pp. 103–108, 2001.
 - [74] G. K. Lee and R. E. Cole, "The Linux Kernel Development As A Model of Open Source Knowledge Creation," Haas School of Business, University of California, Berkeley, Berkeley, CA, Unpublished manuscript December 2000 2000.
 - [75] R. L. Glass, "Of open source, Linux, ...and hype," *IEEE Software*, vol. 16, pp. 126–128, 1999.
 - [76] 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, pp. 309–346, 2002.
 - [77] R. Young, "How Red Hat Software stumbled across a new economy model and helped improve an industry," in *Open sources: voices from the open source revolution*, C. Di Bona, S. Ockman, and M. Stone, Eds. San Francisco: O'Reilly, 1999.
 - [78] B. Behlendorf, "Open source as a business strategy," in *Open sources: Voices from the open source revolution*, C. Di Bona, S. Ockman, and M. Stone, Eds. San Francisco: O'Reilly, 1999.
 - [79] D. Cubranic, "Open-source software development," presented at 2nd Workshop on Software Engineering over the Internet, Los Angeles, 1999.
 - [80] R. A. Guzzo and M. W. Dickson, "Teams in organizations: Recent research on performance effectiveness," *Annual Review of Psychology*, vol. 47, pp. 307–338, 1996.

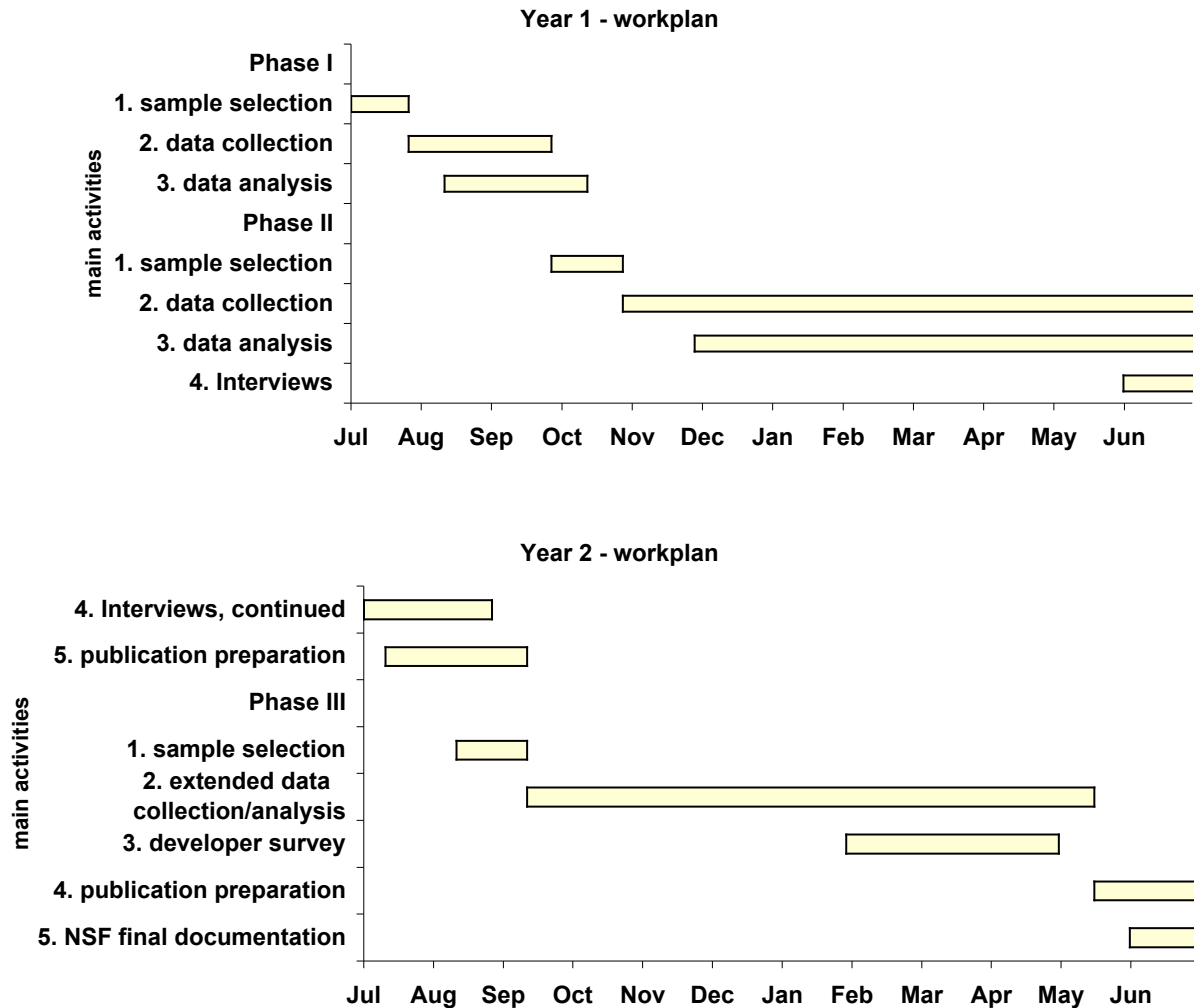
- [81] P. S. Goodman, E. C. Ravlin, and L. Argote, "Current thinking about groups: Setting the stage for new ideas," in *Designing Effective Work Groups*, P. S. Goodman and Associates, Eds. San Francisco, CA: Jossey-Bass, 1986, pp. 1–33.
- [82] H. Kolodny and M. Kiggundu, "Towards the development of a sociotechnical systems model in Woodlands Mechanical Harvesting," *Human Relations*, vol. 33, pp. 623–645, 1980.
- [83] D. Gladstien, "Groups in context: A model of task group effectiveness," *Administrative Science Quarterly*, vol. 29, pp. 499–517, 1984.
- [84] V. F. Nieva, E. A. Fleshman, and A. Rieck, "Team Dimensions: Their Identity, Their Measurement, and Their Relationships," Advanced Research Resources Organizations, Washington, DC, Final Technical Report for Contract No. DAHC19-78-C-0001 1978.
- [85] 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.
- [86] K. Kuwabara, "Linux: A bazaar at the edge of chaos," *First Monday*, vol. 5, 2000.
- [87] R. M. Grant, "Prospering in dynamically-competitive environments: Organizational capability as knowledge integration," *Organizational Science*, vol. 7, pp. 375–387, 1996.
- [88] S. R. Schach, B. Jin, D. R. Wright, G. Z. Heller, and A. J. Offutt, "Maintainability of the Linux Kernel," Department of Electrical Engineering and Computer Science, Vanderbilt University, <http://www.vuse.vanderbilt.edu/%7Esrs/preprints/linux.longitudinal.preprint.pdf>, 2003, accessed 14 Dec 2003.
- [89] S. Faraj and Y. Xiao, "Coordination in fast response organization," presented at Academy of Management Conference, Denver, CO, 2002.
- [90] J. A. Cannon-Bowers and E. Salas, "Reflections on Shared Cognition," *Journal of Organizational Behavior*, vol. 22, pp. 195–202, 2001.
- [91] D. Dougherty, "Interpretive Barriers to Successful Product Innovation in Large Firms," *Organization Science*, vol. 3, pp. 179–202, 1992.
- [92] J. P. Walsh, "Managerial and organizational cognition: Notes from a trip down memory lane," *Organization Science*, vol. 6, pp. 280–321, 1995.
- [93] J. S. Brown and P. Duguid, "Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation," *Organization Science*, vol. 2, pp. 40–57, 1991.
- [94] K. Crowston, "A coordination theory approach to organizational process design," *Organization Science*, vol. 8, pp. 157–175, 1997.
- [95] S. R. Schach, B. Jin, G. Z. Heller, and A. J. Offutt, "Determining the Distribution of Maintenance Categories: Survey versus Empirical Study," <http://www.vuse.vanderbilt.edu/%7Esrs/preprints/lst.preprint.pdf>, 2003, accessed 14 Dec 2003.
- [96] D. Giorgetti and F. Sebastiani, "Automating survey coding by multiclass text categorization techniques," *Journal of the American Society for Information Science and Technology*, vol. 54, pp. 1269–1277, 2003.
- [97] 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, 2003.
- [98] B. J. Dempsey, D. Weiss, P. Jones, and J. Greenberg, "Who is an Open Source Software developer?," *Communications of the ACM*, vol. 45, pp. 67–72, 2002.
- [99] S. O'Mahony and F. Ferraro, "Managing the Boundary of an 'Open' Project," presented at Santa Fe Institute (SFI) Workshop on The Network Construction of Markets, 2003.

- [100] E. Webb and K. E. Weick, "Unobtrusive measures in organizational theory: A reminder," *Administrative Science Quarterly*, vol. 24, pp. 650–659, 1979.
- [101] D. Cubranic, "The ramp-up challenge in open-source software projects," Department of Computer Science, University of British Columbia, Vancouver, BC, Canada n.d.
- [102] 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.
- [103] T. L. Graves, "Inferring Change Effort from Configuration Management Databases," 1998.
- [104] I. Tuomi, "Evolution of the Linux Credits File: Methodological Challenges and Reference Data for Open Source Research," Joint Research Centre, Institute for Prospective Technological Studies, <http://www.jrc.es/~tuomiil/articles/EvolutionOfTheLinuxCreditsFile.pdf>, 2002, accessed 15 November 2002.
- [105] S. Koch and G. Schneider, "Effort, co-operation and co-ordination in an open source software project: GNOME," *Information Systems Journal*, vol. 12, pp. 27–42, 2002.
- [106] E. Chell, "Critical incident technique," in *Qualitative methods and analysis in organizational research: A practical guide*, G. Symon, Ed. London: Sage, 1998, pp. 51–72.
- [107] M. Mortensen and P. Hinds, "Fuzzy teams: Boundary disagreement in distributed and collocated teams," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 284–308.
- [108] S. Weisband, "Maintaining awareness in distributed team collaboration: Implications for leadership and performance," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 311–333.
- [109] B. A. Nardi and S. Whittaker, "The place of face-to-face communication in distributed work," in *Distributed Work*, P. Hinds and S. Kiesler, Eds. Cambridge, MA: MIT Press, 2002, pp. 83–110.
- [110] K. Edwards, "Epistemic Communities, Situated Learning and Open Source Software Development," presented at Epistemic Cultures and the Practice of Interdisciplinarity Workshop, NTNU, Trondheim, 2001.
- [111] M. B. Miles and A. M. Huberman, *Qualitative Data Analysis : An Expanded Sourcebook*, 2nd ed. Thousand Oaks: Sage Publications, 1994.
- [112] R. Heckman and H. Annabi, "A Content Analytic Comparison of FTF and ALN Case-Study Discussions," presented at 36th Annual Hawaii International Conference on System Sciences (HICSS'03), Big Island, Hawaii, 2003.
- [113] A. Edmondson, "Psychological Safety and Learning Behavior in Work Teams," *Administrative Science Quarterly*, vol. 44, pp. 350–383, 1999.
- [114] G. Baker-Brown, E. Ballard, S. Bluck, B. DeVries, P. Suedfeld, and P. Tetlock, "Coding Manual for Conceptual/Integrative Complexity," University of British Columbia and University of California, Berkeley 1990.
- [115] 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.
- [116] A. H. van de Ven and M. S. Poole, "Methods for studying innovation development in the Minnesota Innovations Research Program," *Organization Science*, vol. 1, pp. 313–335, 1990.
- [117] Y. Yamauchi, M. Yokozawa, T. Shinohara, and T. Ishida, "Collaboration with lean media: How open-source software succeeds," in *Proceedings of CSCW'00*. Philadelphia, PA, 2000, pp. 329–338.
- [118] A. Abbott, "A primer on sequence methods," *Organization Science*, vol. 1, pp. 375–392, 1990.

- [119] K. Crowston and C. S. Osborn, "A coordination theory approach to process description and redesign," in *Organizing Business Knowledge: The MIT Process Handbook*, T. W. Malone, K. Crowston, and G. Herman, Eds. Cambridge, MA: MIT Press, 2003.
- [120] K. M. Carley and M. Palmquist, "Extracting, representing and analyzing mental models," *Social Forces*, vol. 70, pp. 601–636, 1992.
- [121] K. M. Carley, "Extracting team mental models through textual analysis," *Journal of Organizational Behaviour*, vol. 18, pp. 533–558, 1997.
- [122] K. Langfield-Smith, "Exploring the need for a shared cognitive map," *Journal of management studies*, vol. 29, pp. 349–368, 1992.
- [123] S. Nadkarni and F. F.-H. Nah, "Aggregated Causal Maps: An Approach To Elicit And Aggregate The Knowledge Of Multiple Experts," *Communications of the Association for Information Systems*, vol. 12, pp. 406–436, 2003.
- [124] V. Albino, P. Pontrandolfo, and B. Scozzi, "Improving innovation projects by an information-based methodology," *International Journal of Automotive Technology and Management*, vol. 3, pp. 249–278, 2003.
- [125] V. Albino, P. Pontrandolfo, and B. Scozzi, "Analysis of information flows to enhance the coordination of production processes," *International Journal of Production Economics*, vol. 75, pp. 7–9, 2002.
- [126] V. Albino, S. Kuhtz, and B. Scozzi, "Actors and cognitive maps on sustainable development in industrial district," presented at Uddevalla Symposium, Uddevalla, Sweden, 2003.
- [127] N. Carbonara and B. Scozzi, "Cognitive maps to analyze new product development processes: A case study," presented at 10th International Product Development Management Conference, Brussels, Belgium, 2003.
- [128] F. Bélanger and R. Collins, "Distributed Work Arrangements: A Research Framework," *The Information Society*, vol. 14, pp. 137–152, 1998.
- [129] E. Carmel and R. Agarwal, "Tactical approaches for alleviating distance in global software development," *IEEE Software*, pp. 22–29, 2001.
- [130] 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*: IEEE Press, 2000, pp. 11 pages.
- [131] K. Crowston and R. Wigand, "Real estate war in cyberspace: An emerging electronic market?," *International Journal of Electronic Markets*, vol. 9, pp. 1–8, 1999.
- [132] K. Crowston, S. Sawyer, and R. Wigand, "Investigating the interplay between structure and technology in the real estate industry," *Information, Technology and People*, vol. 14, pp. 163–183, 2001.
- [133] S. Sawyer, K. Crowston, R. Wigand, and M. Allbritton, "The social embeddedness of transactions: Evidence from the residential real estate industry," *The Information Society*, vol. 19, pp. 135–154, 2003.
- [134] K. Crowston, S. Sawyer, and R. Wigand, "Investigating the interplay between structure and technology in the real estate industry," presented at Organizational Communications and Information Systems Division, Academy of Management Conference, Chicago, IL, 1999.
- [135] K. Crowston and R. Wigand, "Use of the web for electronic commerce in real estate," presented at Association for Information Systems Americas Conference, Baltimore, MD, 1998.

Justification of level of effort

We have divided the two phases of the study into several subphases, as shown in the following workplans.



The principal investigator will work one-third-time on the project during the summers, 1 month per year. Summers will be devoted to sample selection, interviews, final data analysis and publication of results. The PI will devote 10% of effort during the academic year to project management and oversight (1/2 day / week, supported by Syracuse University).

To carry out the project, the PI will involve three graduate students and two undergraduate students. The *two undergraduate students* will be employed for 10 hours/week each during the 30 weeks of the academic year, for a total of about 600 hours (1200 hours in two years). The two undergraduate students will work on development of Perl scripts to download and process data, on data management and on the survey Website.

The *graduate students* will each devote 50% effort during the academic year and 100% effort during the summers, for a total of 3300 hours/year (6600 total hours in two years). Two of the graduate students will support the principal investigator in sample section, definition of constructs and variables, and will

have primary responsibility for data collection and analysis, under the oversight of the PI. They will also have immediate oversight of the work of the undergraduate students. The third graduate student will be assigned to carry out a virtual ethnographic study of project teams. As some analysis will be done in collaboration with Dr. Scozzi, the budget includes travel to Bari, Italy for the PI and graduate students.



Politecnico di Bari
DIPARTIMENTO DI INGEGNERIA MECCANICA E
GESTIONALE

VIALE JAPIGIA 182 - 70126 BARI (BA) - ITALIA

Direzione Tel. 080/5962702 - Amministrazione Tel. 080/5962752,
Fax 080/5962777

Sez. Progettazione: Tel. 080/5962700, Fax 080/5962.777 /741

Sez. Produzione: Tel. 080/5962756, Fax 080/5962788

Sez. Macchine ed Energetica: Tel. 080/5963 480, Fax 080/5963411

dr. Barbara Scozzi

22 December 2003

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

Dear Kevin:

I would like to be involved as a collaborator in your NSF project “Effective work practices for distributed software development”. My participation would represent a chance to advance my understandings of the work practices adopted in a computer-mediated environment, while working in a very stimulating international context. I believe that my contribution would also be extremely useful for the project development.

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

This project would be a good extension to our current research on OSS project’s success factors, some results of which have already been published on an academic journal (please refer to the attached biographical sketch). Our joint research activity develops within a collaborative research agreement established between my department (Department of Mechanics and Business Engineering - Polytechnic of Bari) and the School of Information Studies (Syracuse University). The agreement is aimed at promoting the development of joint research projects, fostering the exchange of scientific knowledge and facilitating the mobility of professors and students. My

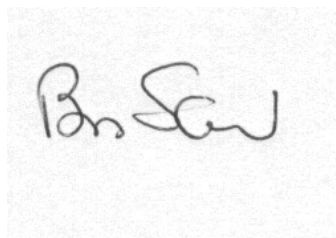
eventual involvement in the project would, thus, be part of a research program already in progress, for which I'm going to present a proposal project to the European Community and the Italian Minister of Instruction, University and Research.

My competencies in the study of coordination and knowledge management practices as well as my interest in cognitive mapping could be particularly useful for the project outcomes. In particular, I would like to contribute to the analysis of the coordination modes adopted within the OSS team, starting from the bug fixing process. I could code, abstract and analyze the data (downloaded from the project's bug tracking systems) related to accomplished tasks, dependencies among them, involved resources, the role they have, and the coordination mechanisms they adopt. The presence of a different view would foster discussion and improve the activity of data comparison and the definition of constructs and variables suitable to characterize the adopted practices. Moreover, this activity could be useful to provide inter-rater reliability in data coding. As I'm also interested in studying the cognitive models adopted within OSS development teams, I could interview (face-to-face and by email) the OSS developers involved in some selected projects so as to develop their cognitive maps. The analysis and comparison among the maps would be aimed at verifying the existence (or the lack of) of a shared perspective about the adopted practices with a specific focus on learning and socialization. Also I would investigate the relationship among the existence (or the lack of) a collective mind and the work performance. Such research hypotheses have been already reported in the section about the research design of the project.

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

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

Barbara Scozzi

A handwritten signature in black ink, appearing to read 'B. Scozzi', on a light-colored background.