PROJECT SUMMARY

Overview:

The proposed study seeks to develop theory and tools to improve the coordination of distributed teamwork with implications for collective work more generally. Distributed teams are groups of geographically dispersed individuals working together over time towards a common goal. Distributed teams are particularly attractive for knowledge-based tasks such as software development because the work can be shared via the same systems used to support team interactions [115, 137]. However, Watson-Manheim, Chudoba and Crowston [154] note that distributed work is characterized by numerous discontinuities that make it harder for team members to work together.

Nevertheless, there can be ways in which distributed work can be better than face-to-face work. In particular, when work products are shared via a computer system, team participants can see the artefacts produced by remote colleagues as easily as from those who are locally situated[59], and these artefacts can provide information to support team coordination. The proposed study builds on prior work suggesting that free/libre open source software(FLOSS) developers use shared artefacts(i.e., the code they are collectively developing) as a basis for coordinating their work, a phenomenon known as stigmergic coordination. This study therefore proposes to address the following research question:
How do distributed teams use shared work artefacts to support coordination of their work? Specifically, what socio-technical affordances enable stigmergic coordination in such teams?

To answer these questions, a two-phase study is proposed: first identifying the socio-technical affordance enabling stigmergic coordination in FLOSS development teams(as exemplars of distributed teams) and similar settings, and second, testing the emerging theoretical understanding by implementing and assessing a system to support stigmergic coordination of distributed work in a new domain.

Keywords:stigermergic coordination;distributed groups;free open source;computer-supported cooperative work

Intellectual Merit :

The study has several expected intellectual contributions. First, the empirical study should provide evidence for (or possibly against) stigmergic coordination as a mode of coordination in distributed work. At present, the evidence for stigmergy is mostly indirect. As well, the study will identify possible negative outcomes from the use of stigmergy. More importantly, the study will identify socio-technical affordances that enable the use of stigmergy. Knowing these affordances will provide a basis for designing shared-work systems that support stigmergy. They will also help in understanding the possibilities and limits on the transfer of coordination mechanisms from open content creation teams to other domains. The proposal includes a plan to study stigmergy based on a theoretical model of coordination and stigmergy and the PI's prior NSF-supported research on FLOSS development processes.

Broader Impacts :

The study has several broader impacts. First, being able to implement a novel mode of coordination could be transformative for the conduct of online work and computer-supported work more generally. As an example of a longer-term indirect benefit, researchers have noted the potential benefit of supporting stigmergic coordination among emergency responders [e.g., 3, 5, 29, 105]. Second, if it seems that reliance on stigmergy is off-putting for certain potential group participants (women in particular), then the project can study how that happens and potentially mitigate the huge gender gap currently observed in FLOSS development participation in particular and in online groups more generally. Third, the software system to be developed will be released as open source for use in future research (thus contributing to the infrastructure for research) and potentially for use by distributed workers (thus potentially benefiting society). Third, the project will provide educational opportunities for a doctoral student and, as well, for students who will use the system as it is developed. Finally, the PI currently advises one minority PhD student and will seek opportunities to involve other members of under-represented groups in the project.

Introduction and problem statement: Supporting stigmergic coordination in distributed teams

The proposed study seeks to develop theory and tools to improve the coordination of distributed teamwork. 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., 117], advances in information and communication technologies have been crucial enablers for recent developments of this organizational form [1] and as a result, distributed teams have become popular [106]. Distributed teams seem particularly attractive for knowledge-based tasks such as software development because the work can be shared via the same systems used to support team interactions [115, 137].

While distributed teams have many potential benefits, distributed workers face many challenges. Watson-Manheim, et al. [154] suggest that distributed work is characterized by numerous discontinuities, that is, a lack of coherence in some aspects of the work setting (e.g., organizational membership, business function, task, language or culture) that hinders members trying to make sense of the task and communication with others [151], or that produces unintended information filtering [61] or misunderstandings [8]. These interpretative difficulties, in turn, make it hard for team members to develop shared mental models of the developing project [54, 70]. The presence of discontinuities seems likely to be particularly problematic for software developers [151], hence our initial interest in distributed software development. Studies of software development teams [53, 92, 136, 151, 153] conclude that system development requires knowledge from many domains, which is thinly spread among different developers [53]. As a result, large projects require a high degree of knowledge integration and the coordinated efforts of developers [19].

In the proposed project, we focus in particular on coordination of distributed work, that is, how team members manage dependencies among tasks. Coordination has been a perennial topic in the study of teams, as well as in empirical software engineering. As Kalliamvakou put it, "To developers, collaboration is equivalent to managing independent contributions to the common whole" [97]. Coordination is clearly important for team effectiveness: for example, Cataldo and Herbsleb [24] found that a failure to match coordination to coordination needs led to an increase in code defects. Distributed work seems to be a particular challenge for coordination. More effort is required for interaction when participants are distant and unfamiliar with each others' work [118, 140]. Considering software engineering again, the additional effort required for distributed work often translates into delays in software release compared to traditional face-to-face teams [85, 110]. These problems are reflected in Conway's law [30], which states that the structure of a product mirrors the structure of the organization that creates it. Accordingly, it would be expected that splitting software development across a distributed team would make achieving an integrated product more difficult [84].

Recent computer-supported team research has adopted a positive perspective, seeking to identify ways that technology use can improve team processes and outcomes, in contrast to a problem-solving approach that seeks to identify problems to address [e.g., 22]. And despite the numerous challenges, there are ways in which distributed work can in fact be better than face-to-face [88]. In particular, when work products are shared via a computer system, team participants can see the artefacts produced by remote colleagues as easily as those from local colleagues [59] and these artefacts can provide information to support team coordination. As we discuss below, coordination through artefacts (the stuff actually worked on, such as software or documents) is different than coordination through prior planning, roles or explicit discussion. We therefore propose a study to address the following general research question:

How do distributed teams use shared work artefacts to support coordination of their work?

To answer this question, we propose a two-phase study, first identifying the ways team members use artefacts to support work coordination in software development and other settings, and especially, the socio-technical affordances of the systems that enable such use. The project will also assess the impact of reliance on this form of coordination on team functioning and participation in distributed work projects. In the second phase, we will test our emerging theoretical understanding by designing and implementing a system to support distributed work in a new domain by supporting the identified affordances and assessing how well they serve to support coordination of team work.

Initial setting: Open source software development teams

Building on the PI's prior work, the initial phase of our study will be set primarily in the context of Free/Libre Open Source Software (FLOSS) development teams, as examples of successful distributed teams with novel approaches to coordination. FLOSS is a broad term used to embrace software developed and released under an "open source" license allowing inspection, modification and redistribution of the software's source without charge. Key to our interest is the fact that most FLOSS software is developed by distributed teams, as developers contribute from around the world, meet face-to-face infrequently if at all, and coordinate their work primarily by means of computer-mediated communications (CMC) [132, 155]. Due to their distributed nature, these teams depend on processes that span traditional boundaries of place and ownership. The research literature on software engineering emphasizes the difficulties of distributed development, but the case of FLOSS presents an intriguing counter-example.

What is perhaps most surprising about FLOSS development processes is that developers appear often to eschew traditional project coordination mechanisms such as formal planning, system-level design, schedules, and defined development processes [84]. As well, many (though by no means all) FLOSS programmers contribute to projects as volunteers, and in most cases, without working for a common organization. Characterized by a globally distributed developer force and a rapid and reliable development process, FLOSS development teams somehow profit from the advantages and overcome the challenges of distributed work [4]. Indeed, the subtext of many FLOSS studies is to understand how those work practices can be applied to other settings.

The growing research literature on FLOSS development has addressed a variety of questions. A complete review is beyond the scope of this proposal: numerous review articles have surveyed research on FLOSS [e.g., 2, 50, 121, 135] and there exist Web sites with collections of articles [143].The most relevant work for this proposal is the research on team work practices and in particular, research on team coordination, which will be reviewed below. The PI on this proposal has been active in FLOSS research, supported by two prior NSF awards. The results of this funding include a review of FLOSS research [50] an analysis of FLOSS teams as virtual organizations [48], models of FLOSS team effectiveness [36, 38] and a study of possible success measures for FLOSS [35, 37]. Empirically, the PI and his team have analyzed the problems in using data from SourceForge [90], carried out social network analyses to understand the centralization and the hierarchy of project teams [39, 40], and described the role of face-to-face meetings in FLOSS teams [41]. Specific processes examined include decision making [82, 102, 103], leadership [69, 83, 109] and group maintenance [6, 138, 139, 156]. A particular focus has been FLOSS team coordination. He and his colleagues have compared the coordination of the FLOSS bug-fixing process to propriety software [32, 49], found self-assignment of work to be a common task-assignment approach [45, 51] and shown how the motivational features of FLOSS development lead to a layered structure for the software [91].

These earlier grants were aimed at identifying work practices that characterize effective FLOSS teams and the dynamics of their operation. In the research proposed here, we will carry out a more focused study of coordination practices in FLOSS teams, compare these processes to those in other kinds of distributed teams and then apply the findings to a novel domain. We have chosen this focus because studies of FLOSS teams (including our own) and of distributed teams more generally point to the possibility of a novel form of coordination (stigmergy, described below) that needs more research to fully understand but which could be useful if it could be generalized.

The "coordination paradox"— How can distributed teams coordinate without communicating?

In this section, we draw on our prior work on FLOSS coordination to describe the paradox that motivates this proposal: the apparent ability of teams to coordinate with little or no explicit communication. This finding emerged from a study of how FLOSS developers coordinate their work [14, 89, 91]. Somewhat unexpectedly, in the study we found little evidence of overt coordination of the development, i.e., FLOSS developers seemed to rarely communicate about coding tasks. The lack of evidence was surprising considering the transparency of FLOSS projects: we expected to find direct, discursive communication in email or other discussion fora through which developers interact . But we found few examples. The lack of direct interaction around the work has echoes in our other research

findings. For example, we found that developers mostly self-assign work rather than have it assigned to them [45, 51] and often make decisions about code without explicitly evaluating options [81, 82]. Interestingly, when developers do discuss their work, they often refer directly to the software code.

One interpretation of these findings is that rather than coordinating explicitly, FLOSS developers rely instead on implicit coordination [134], e.g., by sharing well-developed mental models [43, 71, 72] or shared understandings [17] of the task that allow them to determine what needs to be done without the need for explicit communication and coordination. However, while developers clearly have and rely on mental models of the task, it seems unlikely that these explain the paradox by themselves. First, the FLOSS development process is highly complex and ever changing. It seems impossible that developers can keep their mental models up to date given the ever-changing dependencies within the code and modifications made by numerous other developers. Second, the problem of coordination is exacerbated as the participation of developers waxes and wanes over time. FLOSS participants are not all experts but range from newcomers to experienced software engineers. For implicit coordination to be sufficient, we would need to explain how inexperienced participants develop mental models sufficiently robust enough to address the coordination needs. In short, while implicit coordination is important, this mechanism is not sufficient by itself.

In this proposal, we offer a complementary perspective on coordination. We focus here on the evidence presented above that on the infrequent occasions when they do interact, developers often refer to the code that they are collectively developing. We theorize that work can be coordinated through the outcome of the work itself, a mode of coordination analogous to the biological process of stigmergy [78]. As Heylighen writes, "A process is stigmergic if the work…done by one agent provides a stimulus ('stigma') that entices other agents to continue the job" [86]. The question then is how the work can support such coordination. From this perspective, we state a more specific research question:

What socio-technical affordances of shared work systems enable stigmergic coordination?

By socio-technical affordances, we mean the features of the technology used and the practices around that technology. For example, the source code control systems commonly used by FLOSS developers provide notifications of code submissions; details of the implementation of this technical feature enable other developers to maintain awareness of the state of the code to support coordination. To interpret these change messages, developers likely need some level of technical skill and mental models of the code structure, another kind of affordance. They may also be accustomed to creating code in a way that is easier for others to interpret. The inherent nature of the coding task itself may create the need for specific kinds of coordination that are particularly amenable to stigmergy.

It is important to note that we are not arguing that stigmergic coordination completely replaces other forms of coordination. We rather see these different modes as complementary. Developers clearly still need to talk on occasion and there seems to be an important role for shared mental models in being able to interpret the stigma to guide action. Kalliamvakou quoted FLOSS developers as saying "because the developers are speaking the same language it is easier," and "members are familiar with the idea of working this way and share the mentality behind it" [97].

Literature review

In this section, we review the theoretical framework and related work that inform our study.

Coordination theory

We first introduce the topic of coordination and present the fundamentals of coordination theory, the theoretical foundation for the proposed study. Coordination theory [104] synthesizes the contributions proposed in different disciplines to develop a systemic approach to the study of coordination. In this perspective, studying coordination means analyzing the management of the dependencies that emerge among the tasks and components of a system. This definition of coordination is consistent with the large body of literature developed in the field of organization theory [e.g., 74, 101, 108, 126, 149] that emphasizes the importance of interdependence in group work.

Malone and Crowston [104] analyzed group action in terms of actors performing interdependent tasks to achieve some goal (i.e., in an organizational process [31, 46]). These tasks might require or create

resources of various types. For example, in the case of software development, actors include the user and various members of the software development team. Tasks include translating aspects of a user's problem into system requirements and code, or bug reports into bug fixes. Resources include information about the users' problems and developers' time and effort.

Coordination theory defines coordination as "managing dependencies" and conceptualizes dependencies as arising between multiple tasks when the tasks use or create the same resources. Dependencies come in three kinds. First, flow dependencies match Thompson's sequential dependency [149]: one task creates a resource that a second uses. Flow dependencies create the need to manage the usability of the resource and the timing and location of its availability. Second, a fit dependency occurs when the output of two tasks must fit together in the creation of a common resource. Alternately, if the output of the two tasks is identical, there is potential synergy, as the duplicate work can be avoided. Finally, a shared resource dependency emerges among tasks that use a common resource (like Thompson's pooled dependency). Resources may also be directly interdependent, e.g., due to physical connections, in which case there can be dependencies between the tasks that use connected resources.

The key point in coordination theory is that dependencies create problems or potential synergies that require additional work to manage. Malone and Crowston [104] called the tasks embodying this extra work coordination mechanisms. For example, if particular expertise is necessary to perform a given task (a task-actor dependency), then an actor with that expertise must be identified and the task assigned to him or her. There are often several coordination mechanisms that can be used to manage a given dependency. For example, mechanisms to manage the dependency between a task and an actor include (among others): (1) having a manager pick an appropriate subordinate to perform the task; (2) assigning the task to the first available actor, regardless of skill; (3) a labour market in which actors bid on tasks; and (4) self-assignment of tasks based on individual interest, as in FLOSS. To manage a usability subdependency, the resource might be tailored to the needs of the consumer (meaning that the consumer has to provide that information to the producer) or a producer might produce to a standard so the consumer knows what resource to expect. To manage shared use of a resource, tasks might take turns, first-come-first-served or be given a reserved time slot in which to use the resource.

All collaborative work requires some coordination and so coordination mechanisms may be useful in a wide variety of organizational settings. Conversely, organizations with similar goals achieved via the same set of tasks will have to manage the same dependencies, but may choose different coordination mechanisms, resulting in different processes. And the mechanisms are themselves tasks that must be performed by some actors, so adding coordination mechanisms to a process may create additional dependences that must in turn be managed.

As an example of a coordination theory analysis, we can identify numerous dependencies in the software development process that need to be managed, implying the need for matching coordination mechanisms. Since developers work on the same codebase, there is a dependency between their work, requiring mechanisms for resource sharing to avoid conflicting changes. For example, Blincoe, Valetto and Goggins [12], [13] used traces of developers' work on the same files to identify the potential need for two developers to coordinate. An important dependency is between a task (e.g., a bug report) needing to be done and someone to work on it, requiring mechanisms for task assignment (e.g., the bug might be assigned by a development manager to a developer to ensure that it is fixed by exactly one developer). Source code, can be managed proactively (developers check out the code they want to work on, preventing others from making conflicting changes) or optimistically (e.g., change made are checked for conflicts and any detected are resolved). And finally, the code itself has many interdependencies, e.g., a function that calls other functions or two that use shared data. Changing one piece of code can affect these relationships (e.g., changing a function will require changing all places that call that function), requiring special attention when making changes.

We note that the coordination theory framework makes a distinction between the tasks and the mechanisms needed to coordinate the tasks. These two concepts are sometimes labeled "work" versus "articulation work" [75, 144]. The conceptual split between work and coordination of work is also clear in the software engineering literature from Conway [30] through Cataldo and Herbsleb [23]. The duality

between work and coordination arises in part from an information-processing perspective on the work that assumes an input-process-output model of the work, making it natural to consider the tasks that create the output (connecting inputs to outputs) as the main part of the process and coordination mechanisms as separate from this work. Perhaps as a result, much of the focus of research on supporting coordination has addressed ways to improve explicit coordination. For example, an early CSCW system, the "Coordinator", sought to improve coordination by making communication more explicit about the coordination required [73, 167].

Stigmergic coordination

In contrast to the prior focus on explicit coordination, in this study we are interested in how the work itself can serve as guidance for coordination. For this analysis we draw on the biological process of stigmergy [63], defined as a process by which one individual affects the behaviour of others through changes in the shared environment. For example, ants follow scent trails to food found by other ants, thus assigning labour to the most promising sources. But the organized collective action emerges from the interaction of the individuals and the evolving environment, rather than from a shared plan.

While stigmergy was formulated to explain the behaviour of social insects following simple behavioural rules, it has also been invoked to explain classes of human behaviours: the formation of trails in a field as people follow paths initially laid down by others (similar to ant trails), or markets, as buyers and sellers interact through price signals [125]. For humans and intelligent systems, the signs and processing can be more sophisticated than is found for insects [133]. For example, the shared environment can be a complex workspace including annotations. Tummolini and Castelfranchi [150] developed a typology of different kinds of messages possible from signs, such as having the ability to do something, having done something or having a goal. Christensen [25, 27, 28] discussed how architects and builders coordinate their tasks through "the material field of work" such as drawings.

Stigmergy has been suggested in particular as an interpretation of how FLOSS developers coordinate, what Kalliamvakou called a "code-centric collaboration" perspective [97]. FLOSS developers mostly work with the code that they are developing and source code control systems such as Git provide status about the state of the code and development. Dalle and David [60] present a simulation of FLOSS developers allocating work based on information they get from the code base, providing evidence that stigmergy could be a viable approach to coordination in this setting. Stigmergy has also been argued as a mechanism in online work more generally. Elliot [65] argued that "[c]ollaboration in large groups is dependent on stigmergy," with the specific example of authoring on Wikis.

Stigmergy can be readily interpreted in the coordination theory framework developed above. Malone and Crowston [104] describe coordination mechanisms as relying on other necessary group functions, including decision making, communications, and development of shared understandings and collective sense making [18, 43]. The stigmergic approach suggests that the "shared material" itself can be a communications medium, allowing coordination without recourse to separate coordinative mechanisms [27]. Christensen observed this type of coordination among architects, noting that their work is "partly coordinated directly through the material field of work…in addition to relying on second order coordinative efforts (at meetings, over the phone, in emails, in schedules, etc.), actors coordinate and integrate their cooperative efforts by acting directly on the physical traces of work previously accomplished by themselves or others" [26].

The stigmergic perspective can be seen in the context of ongoing debates about the nature of socio-material structures for articulating the entwined nature of work and coordination [123]. Stigmergy resonates with newer theoretical perspectives on human activity that "recognise the importance of the environment", such as "activity theory, situated action, and distributed cognition" [148]. A common element in these perspectives is the description of the role of artefacts in collaboration, i.e., what we are labelling as stigmergy. Stigmergy is also compatible with a structurational perspective on work, which the PI used as a framework for prior studies of FLOSS dynamics. Structuration theory [77] is a broad sociological theory that seeks to unite action and structure. The theory is premised on the duality of structures, meaning that the structural properties of a social system are both the means and the ends of the practices that constitute the social system. Jacob [96] similarly argued that "stigmergic structures inform

and control individual cognitive activities, they are themselves the residual products of successful problem-solving activities undertaken by the larger socio-cultural community".

Studies of stigmergy can also be informed by research on other concepts of long-standing interest in CSCW. First, there has been a stream of research in CSCW and elsewhere that demonstrates the importance of team member awareness for supporting collaborative work. Though they are not identical, there is clearly a close relationship between the two ideas about supporting collaboration. Christensen [27] described actions a person might take to make a co-worker aware of an issue, and so distinguishes awareness from stigmergy, as "stigmergy does not entail making a distinction between the work and extra activities aimed solely at coordinating the work". Similarly, in contrast to active awareness (one participant calling for the attention of another), Dourish and Bellotti [64] argued for the importance of passive awareness mechanisms, which could be interpreted as supporting stigmergy. Other researchers have proposed awareness displays that allow a team member to develop an awareness of the actions of other team members. Carroll and colleagues [20, 21] examine in particular how awareness can support development of common ground, community of practice, social capital and human development in team. In this proposal, we focus more narrowly on how awareness of work supports coordination.

A second related concept is system translucency [68] or transparency [58, 59, 145], meaning visibility of details of organizational processes or functions. Consistent with our analysis of stigmergy, Stuart, et al. [145] analyze transparency as a form of information exchange or communication. They note that technology enables new forms of transparency, e.g., as in GitHub, a software development site [57] that provides real-time updates on what other developers are doing. In other words, transparency is a system feature that might support awareness. Researchers have noted similar problems with awareness and transparency, such as the potential for information overload from having to review too much information or that making too much visible may inhibit the willingness to share work [11, 58].

As with stigmergy, system transparency provides information that can influence how people work. Dabbish, et al. [59] note specifically that transparency is helpful for coordination. They list numerous uses of visibility information, such as including dependencies with other projects [58]. They further note that being able to see something means "much less need for routine technical communication" [58], suggesting that transparency is substituting for explicit coordination. Research on visibility and transparency can clearly be quite informative for designing systems to support stigmergic coordination. However, this stream of research has not specifically focused on the socio-technical affordances that enable users to make sense of and to use the provided stigma to support coordination, which is the goal of the current proposal. For example, given the large number of possible signs available, how do developers decide which to attend to?

A third related concept in the CSCW literature is provenance, i.e., the history of a piece of information. Rather than being explicitly and independently created, provenance of documents is built as the documents are changed, or recorded from interaction as the documents are used, i.e., it is a kind of stigma. Hill, Hollan, Wroblewski and McCandless [87] and Wexelblat and Maes [157] pointed out that knowing how others have interacted with a piece of information can be informative for future interactions with it. Similarly, knowing the history of a document's development is important in evaluating and knowing how to use it.

There are of course many, many collaborative systems designed to support groups and in particular to support coordination of group work (that is, for managing dependencies among group tasks). However, only a few systems have been explicitly aimed at supporting stigmergic coordination. Musil, Musil and Biffl [113] proposed the concept of a Stigmergic Information System (SIS) architecture metamodel, though their goal is to develop an architectural model that describes many different kinds of systems rather than to build one. Most of the systems described as stigmergic appear so far to focus on simply providing access to the shared work, without specific attention to coordination of the work. For example, Zhang, Zhao, Jiang and Jin [173] described a system for allowing collective construction of a conceptual model. Secretan [141] described the Picbreeder system in which users interact only via images to create new images. The proposed work will advance the state of the art by more explicitly addressing how shared work (rather than communication for explicit coordination) supports coordination within a team.

Theory building

In this section, we build an initial theory of the socio-technical affordances of systems necessary to support stigmergic coordination, with particular attention to the relation between stigmergy and other forms of coordination. It is hypothesized that these characteristics of systems for sharing work will support coordination of the work, thus distinguishing a system for stigmergic coordination from systems for explicit coordination on the one hand and systems for simple information sharing on the other.

Documents

To theorize what affordances of work support coordination, we turn to the literature on documents and work [124]. Software code is a semiotic product recorded on a perennial substrate that is endowed with specific attributes intended to facilitate specific practices [172], thus making it a kind of document. Code differs from other kinds of documents by serving two audiences, one being a machine, the other software developers. However, we focus on the latter, describing properties of code that allow developers to share their work with colleagues, and to read, understand and respond to their intentions.

Scholars have described how documentation and other accounts of work play a central role in the coordination of work [15, 16, 142, 146, 147];Østerlund, 2008 #27686;Østerlund, 2008 #23487;Østerlund, 2007 #23488}. These perspectives have long pointed to the double role of documents as both "models of" work and "models for" work. For the first, documents provide an account "of" reality as workers manipulate text and other symbolic structures so as to parallel them with reality. For example, developers may carefully document the code they have constructed to create a report of the work done. But documents also provide a basis from which people further manipulate the world. For example, developers' reports are not simply accounts of work completed: the reports guide ongoing work by prescribing what is left to be done or enabling collaborators to coordinate their work. Taking inspiration from Smith [142] and Bakhtin [9], we suggest that a work product is rarely completely original; it is always an answer (i.e., a response) to work that precedes it, and is therefore always conditioned by, and in turn qualifies, the prior work. What the programmer does when facing somebody's work is responsive and partially determined by what has been going on up until now. The reports are thus accounts "for reality" as they provide a blueprint of the software taking shape. Documents in this way offer a double accountability: when documenting the coding of a software program, developers mold the account to the reality of the code on their computers and at the same time, mold their ongoing coding to the account.

Three further concepts from document studies stand out as helpful in articulating how work can serve as a model for work: genre, visibility and mobility, and combinability. We address these in turn.

Document, genre and genre systems

First, people can recognize a document as a model for possible action only because they have some background knowledge about the genre of that document, and thus the expectations associated with that type of communication [122]. A genre is defined as typified action invoked in response to a recurrent situation [171]. For example, common document genres related to the present proposal include project summary and description, biographical sketch, review, and panel summary. Each has a characteristic form (e.g., a panel template) and purpose. People engage genres to accomplish social actions in particular situations, which are characterized by a particular purpose, content, form, time, place and set of participants.

The same can be said about FLOSS work products. A developer engages in typified actions invoked in response to recurrent situations. They do so to accomplish something characterized by a particular purpose, material form, place, time and participants. By completing a piece of code and leaving it for a colleague to work on, a developer invokes a specific genre of work. The colleague will be able to pick up and work with the code because it invokes that genre and so comes with certain expectations. The first engineer might have created a scaffold of a module that simply outlines a structure. In so doing, his work product becomes a model for work associated with specific elements and course of action. It might invoke a sequence of steps or routes to a conclusion. It might invoke certain categories or socio-material arrangements that will have to be used. In this way, a piece of completed work serves as a model for future work by drawing on its own genre, i.e., what are the expected outcomes, what materials and forms should be invoked at what places and times and by what types of participants.

Furthermore, documents related to work (and so we argue, the work itself) are often organized into what are called genre systems [119], formalized sequences of documents of particular genres providing more or less standardized methods for recognizing what might be done and what does get done as legitimate work. We alluded above to the genre system around NSF proposals: proposal, review and panel summary. The process of publishing a journal paper similarly involves a sequence of documents of specified genres: submission, reviews, editor's report, decision letter, revision, acceptance letter, final submission, galley proof, copyright release and published paper.

A key point in the analysis of work in terms of genres is that for genres and genre systems to enable documents to function as models for work they must be part of the conventions of practice shared among members of particular communities. Genres are not naturally occurring. They are rather learned as part of membership of such communities: As new participants are socialized into the communities, they gradually acquire a naturalized familiarity with the socio-material arrangements and prominent genres.

Turning back to FLOSS, source code provides genre expectations and thus serves as a "model for" work at two levels. First, the FLOSS development process includes a number of distinct and typified actions involved in response to a recurrent situation and expressed in a set of characteristic documents, including source code, bug reports, commit messages and so on. These genres are associated with particular purposes, forms, content, times, places and participants for the related tasks. For example, the purpose of the code is to instruct the machine to perform an application, whereas bug reports are used to provide information to developers about observed problems with a program. Code is used nearly exclusively by developers, while bug reports are shared between users and developers. By looking at these work outputs, experienced FLOSS participants can tell which tasks are called for.

Second, the source code itself has a structure in which each component has more-or-less well-defined purposes associated with particular functionalities. There are genres of source code: It collectively has the purpose of providing instructions for the computer, but as well, each module of a program has its own particular purpose and so its own subgenre. For example, some modules may manage the interface, while others deal with interactions among particular data sources. Clarity of communicative purpose is of critical importance for developers; it should be clear which components are appropriate to modify to add some desired new functionality. In a well-structured program, the purpose of each module is clear—the subgenre is recognizable—and so the code is useable by others as a model for work. In poorly structured code, the purpose of particular module may be hard to determine or, in fact, muddled and unclear. This confusion may not directly affect the functionality of the program, but in these cases, the code does not constitute a genre. Future programmers cannot tell where to add new functionality because the current work outcomes do not make it clear how to add it without negatively interfering with existing functionality. Consistent with this view, developers reflecting on success factors for FLOSS argue that developing the right program structure, one that communicates well to other developers, is key to the success of a FLOSS project.

FLOSS development further provides expectations about sequential ordering of documents. The broad genres of software development seem to be arranged in an organized manner where one task typically follows another. For instance, a developer would first read a bug report, then change the code and commit, creating a source code control system commit message. Releases have their own sequence of documents, such as release note, packaged software, binary distributions and so on. In other words, the FLOSS infrastructure supports certain sequences of documents that suggest particular sequences of processes that participants learn as part of membership in the group.

Visibility and mobility

The second key feature of documents is their visibility and mobility. In order for a piece of work to serve as a model for future actions it must be visible and accessible to others. Obvious as it may seem, making work visible is not a straightforward process. As discussed by Suchman [147], some work may be more visible than other work; some work may cover up previous activity and render it invisible. For example, service work is notoriously hard to make visible: The better such work is done, the less visible it is to those who benefit from it. CSCW research on awareness and transparency also addresses these concerns. Understanding what elements of work are accessible and how its visibility may change over

time is central to understanding how work may or may not serve as a model for future work. Similarly, for work to coordinate tasks beyond a physically-restricted space, it must become mobile [100], meaning that it is conveyed to a context in which others can encounter it.

Most obviously, the FLOSS development infrastructures support the mobility of work by being Web-based. Any FLOSS developer can download the source code from the source code control system and have access to others' work as a basis on which they can build their own. As a result, software engineers can, in many situations, use others' work as a model for their own work because of their ubiquitous access to the server containing the code. Further, many system provide a mechanism to push changes to locally other developers' workspaces, rather than having to wait for those others to seek them out. By being in multiple places, code can coordinate work in multiple settings. In FLOSS, the source code control system also records a revision history: all changes made to each file in the system including what files are created or deleted by whom, when (the file's provenance). Many changes include short notes that can explain why a change was made (although many changes do not, apparently expecting the reader to examine the code directly). Such histories not only serve as 'models of' work but can also point forward by depicting the generally accepted work process. For a newcomer, such histories provide a window to how things are done, what tasks tend to follow what tasks and what is regarded as good and opposed to bad (i.e., reverted) work. Visibility of FLOSS work is promoted as well through cultural norms about development. A widely acknowledged culture norm in open source is to "check in early, and check in often." If people do not share their work often, they are not making it visible to other participants to build on. Contrariwise, large infrequent commits ("code bombs") increase the chances that there will be conflicts and make it harder for other developers to understand what a change does, again hampering visibility. Indeed, a frequent complaint about a code contribution is that it is too large for developers to easily understand.

Combinability

The third important characteristic of work is combinability. For work to be a model for future work, the work must be combinable and improvable in modular increments [91, 100]. If a piece of work is done, nothing is left to do, hence Raymond's surprising injunction to "leave low hanging fruit" [132]. Most work tasks are layered and complex: New work contributions can be adjusted and added to existing outcomes. A piece of work might start out as an incomplete frame, a scaffold on which other parts get added in some organized sequence. Later new functionality can be added to the existing structure. In this way, a program evolves from version to version.

Combinability in FLOSS development is supported both by the source code control system infrastructure and cultural norms. First, there are strong cultural norms for providing "atomic commits," that is, developers are encouraged to address only one change or topic when committing new code, leading to many small commits [7]. It is easier to combine code with a focused commit than with a commit that does multiple things and touches bits and pieces of dozens of files in the process. It is likewise easier to back out a focused commit if things should go wrong. Developers are also warned: "Don't break the trunk", which means that the main set of files in the source code control system should always compile and run. This practice ensures that any developer who downloads the code will be able to work with it, supporting the individual development described above.

Combinability is further supported by the source code control system infrastructure allowing participants to try out experiments on the code in a branch before committing it. Developers can execute and test ideas at any time without interfering with others; they can run the software with their proposed changes and obtain direct feedback about the combinability and thus success or failure of their changes. This approach allows them to iteratively enhance their understanding of the task and to modify their strategy for managing dependencies between the existing system and what they are trying to accomplish. In this way, developers can interact with the code base as they would engage in a conversation by continuously receiving feedback on their output. As a result, developers can avoid a lot of communication with co-developers, since their active engagement with the artefact provides substantial insights; one has less need to ask another what their intentions were when one can experiment with the codebase.

Summary

In summary, from the literature we have an understanding of why coordination is important and a theoretical framework for examining stigmergic coordination in teams. Task design determines what dependencies exist in the work and the coordination mechanisms needed. The literature also provides a starting point for identifying the socio-technical affordances that enable team members to make use of stigmergic coordination, e.g., the role of genres of work and the visibility and combinability of the work contributions. Again, the nature of the tasks will determine the genres of work that provide information to support coordination. At an individual level, prior work suggests that motivation to cooperate will be important [56]. However, further research is needed to clarify and test these relationships.

Plan of work

In this section, we describe our specific plans to identify and test the socio-technical affordances that support stigmergic coordination and how we will evaluate the project.

Study design

In this section, we first present the design of the proposed study, deferring details of data collection and analysis to the following section.

Phase I. Phase I has the primary goal of assessing the importance of stigmergic coordination and identifying the socio-technical affordances of shared work that supports coordination of work in one exemplary setting, namely FLOSS development. From the literature reviewed above, we have developed a theory of what characteristics of work are needed, so some might argue that this phase is unnecessary. However, the evidence for stigmergy is indirect: an absence of visible coordination rather than direct evidence for stigmergy. Therefore, it is necessary to carry out a rigorous study to confirm (or refute) that stigmergy is a viable approach to coordination. We are fairly confident that the "coordination paradox" is a real phenomenon, but if it turns out that FLOSS developers do not rely on stigmergy, it will be valuable to identify the actual mechanism that enables coordination without communication (e.g., communication through some medium other than the work itself). It will also be an intellectual contribution to determine what information developers use as a basis for making decisions related to coordination and the affordances that enable them to obtain and make sense of this information. The project thus extends prior work on visibility and awareness.

The second goal of Phase I is to extend prior work by comparing stigmergy in FLOSS development to other work settings. This goal will be achieved through a set of mini–case studies in diverse domains that exhibit some level of stigmergic coordination, e.g., Christensen[25, 27, 28]'s studies of architects and builders [25, 27, 28]. A particular domain of interest is Wikipedia editing, as creation of Wikipedia articles has many parallels to creation of FLOSS (as well as important differences). Elliot [65] suggested that authoring on Wikis is guided by stigmergy. More specifically, den Besten, Gaio, Rossi and Dalle [62] examined the use of tags in Wikipedia to direct actions of editors and found that the application of tags to articles was associated with change in editor behaviour, which they interpret as evidence for stigmergic coordination. More generally, collaborative writing [10] requires coordination that can be done in part through the document itself. Selection of cases will be based on theoretical sampling, guided initially by the literature on stigmergy briefly reviewed above. Completing six to eight cases should be feasible and still provide enough richness for comparison.

Given that several studies have already identified the possibilities of stigmergic coordination, the mini-cases can started based on a literature review. However, they will have to go beyond simply identifying the possibility for stigmergy, the focus of most current published reports, in order to identify the socio-technical affordances that support stigmergic coordination in these different domains.

These cases will also help identify the limits of stigmergy. For example, the high level of discussion around Wikipedia articles [e.g., 99, 120, 152] suggests that coordination in this setting is only partly stigmergic. A comparative analysis can identify kinds of dependencies and coordination mechanisms that are not supported by stigmergy.

A final goal of this phase of the study is to assess the impacts of a reliance on stigmergic coordination on teams. To conceptualize these impacts, we draw on the input-moderators-output-inputs (IMOI) model [93] used by the PI in a prior study of FLOSS development [50, 93]. For example, the use of stigmergic

coordination instead of explicit communication (a team process) may allow decisions to be made more quickly by obviating the need for consultation. Carroll and colleagues [20, 21] suggested that team awareness can support development of social capital in teams, but a lack of conversation might instead lead to diminished social capital (a team emergent state), harming the long-term functioning of the team. It may also be that a setting with minimal direct communication has a differential impact on interest in team participation. Specifically, a strong concern about FLOSS development as a model for virtual technology work is the extraordinary lack of diversity among developers of these projects. In studies of SourceForge, MySQL and OpenOffice discussion fora, researchers found only 4% of respondents were female [98], a level comparable to earlier findings [76]. While the problem of differential participation is multi-faceted, part of the problem appears to be that the sociocultural setting and practices of FLOSS work are less attractive to women [114], which could be due in part to the reliance on stigmergy (among many other factors). The PI did early work on gender differences in the use of computer-mediated communications that will inform this aspect of the study [42].

Phase II. In Phase II, we will implement a collaborative shared-work system for a novel domain that supports the affordances identified in phase I as a way to test our understanding of the role and importance of affordances.

During the summer between years 1 and 2, we will examine varied domains (e.g., starting with the mini-case studies) to identify likely settings for testing the hypothesized socio-technical affordances. At present, we are considering software analysis and design, i.e., by supporting distributed development of UML diagrams. System analysis and design has the advantage of being close to software development, making it more plausible that stigmergy can be transferred to this domain. On the other hand, the reliance on graphical rather than textual documents will pose challenges in other areas, e.g., combinability. The final choice of domain will be guided by the findings of Phase I regarding necessary affordances and possible domains for stigmergy. A consideration is that the results of the study will be more valuable if they can be transferred to a task that is often carried out in distributed settings.

Having chosen a domain, we will next determine how to transfer the identified affordances. Based on the findings of Phase I, we will identify system features to support stigmergic coordination in the selected domain, along with social features needed to make these technical features feasible. For example, a prominent feature of FLOSS is the use of source code control systems that provide notifications of changes and bundle related changes to make them easier to review. However, such messages are likely informative only to developers who already understand the structure of the system and who can therefore quickly apprehend the import of the changes for their own work.

We plan to adopt a design approach that mixes system development and assessment of use. An initial prototype will be developed based on the results of Phase I, but as early as possible, we will have users interact with the system to help us understand how it is functioning and where changes are needed. It will likely be informative to observe use of the unmodified tool to obtain baseline data (e.g., how developers coordinate using an unmodified version of a drawing tool); these observations can be started as one of the case studies and continued during the first six months of year 2 while an initial system is implemented. The goal is of the system assessment is not simply to demonstrate that the new tool works, but rather to give us rich information about how people are using the newly-provided features in order to iterate the design. Experience with the tool will inform the theorizing and provide a test of our understanding. We will make the developed system available so future research can examine usefulness more generally.

Data collection and system development

In this section, we describe the planned details of data collection and analysis as well as plans for system development.

Phase I. For Phase I, the primary source of data will be interviews with key informants in FLOSS projects. FLOSS teams are of varying sizes, with different compositions in terms of level of participation and geographical distribution. The need for coordination will be most pronounced in larger teams working on larger and more complicated systems. Communication challenges will be most pronounced in teams with distributed members that lack frequent opportunities to communicate. As well, more active projects will produce more code changes that increase the need for awareness of what other members are

doing. Therefore, to maximize the chance of seeing stigmergic coordination in action, we will focus our attention on large, complex, active projects with high levels of distribution and heterogeneity in participation. These features describe many of the Apache Software Foundation projects, for example.

Interviews will be conducted mainly by phone, Skype or e-mail, but we also plan to attend one or two FLOSS conferences each year (e.g., *ApacheCon*) to interview developers face-to-face (as we did with good results in prior studies). Dabbish, et al. [59] interviewed 24 developers in their study, about the same number that we interviewed for our prior studies. Our past experience suggests that we should be able to recruit that many FLOSS developers for interviews. To complete initial data collection over the winter of year 1 requires interviewing two to three developers a week, an aggressive but feasible schedule that leaves time for analysis and follow-up interviews.

We have carried out a small pilot study of stigmergy in FLOSS development for which we developed an initial interview protocol. The protocol is structured along the elements of the coordination theory framework, e.g., with questions about how decisions are made, what kinds of dependencies are problematic and what coordination mechanisms are employed. Specifically, it includes questions about the interviewee's background, description of the FLOSS project worked on, the model of collaboration, tools used, work practices, how the work is coordinated, and what kind of information is obtained from the shared source code to guide decisions about the subject's work.

Subjects interviewed in the pilot contributed to small projects that did not face high needs for coordination, so the data are not very conclusive on the role of stigmergic coordination. However, through the pilot we were able to refine the questions about their work and coordination mechanisms. As well, we have an initial list of tools used to support group collaboration, e.g., GitHub and Bitbucket for code hosting, Slack for communication, and Trello and Pivotal Tracker for issue tracking. Of these, the first two could support stigmergic coordination while the others support communications or explicit coordination. However, we still have much work to do to understand the socio-technical affordances that allow developers to extract necessary information from shared code.

For the proposed study, the protocol will be augmented to elicit further data to address the theoretical model developed above. Drawing on the PI's prior work on genres [44, 52], we will identify prominent genres of information used and their role in genre systems. To assess visibility and mobility, we will identify how work products are accessed, examine the gap between production and use and check whether subjects can say where a particular piece of work came from. To assess combinability, we will examine how work provided by others is incorporated into the subject's own work. We will probe for evidence of shared mental models that support implicit coordination or that support the identification and use of different genres. The interview protocol will also include open-ended questions to probe other ways in which shared work or other sources of information are used as a guide for the subject's own work. As part of the interviews, we will employ the critical incident technique, in which developers are asked to describe personally experienced specific incidents that had an important effect on the coordination to understand the factors involved. We plan to follow the approach adopted by [59], in which we ask developers to talk about how they use systems, what information they get and how they use it. Interviews will be recorded when possible and transcribed for analysis.

We also plan to examine developer email mailing lists for mentions of looking at source code as a basis for making decisions about coordination. We will work with the PI's SOCQA project (described below under results from prior NSF funding) to explore the use of natural language processing techniques to automate this analysis, in order to analyze a sufficient volume of text to be able to draw conclusions.

Data will be content analyzed following the process suggested by Miles and Huberman [107], iterating between data collection, data reduction (coding), data display, and drawing and verifying conclusions. The researchers will develop an initial content analytic framework to uncover the patterns of the concepts present in the data. The initial (deductive) framework will be based on the concepts in the literature review. As the analysis continues and new concepts emerge, they will be added to the framework and to the interview protocols. The interviews will also add to the inventory of systems used in the groups, which can be mined for relevant features that support coordination, including, in particular, features for making work visible and thus supporting stigmergic coordination in this setting.

The short case studies (goal two of Phase I) will be developed from the published literature, interviews with a small number of participants and examination of artefacts that are group created, evolved and maintained in those settings. Interviews will use an adapted version of the interview protocol and analysis approach described above. For example, interviews will be done with a few Wikipedia editors to understand how the evolving articles themselves guide editors' work. As well, we have agreement from other researchers at Syracuse studying distributed teams to look for evidence of stigmergic coordination in their settings, i.e., distributed science teams and biohackers. In these settings, we can look for evidence of stigmergic coordination among amateur makers and science teams supported by shared artefacts including drawings, schematics, circuit design documents, user studies and interviews.

Finally, to address the third goal of Phase I, we will employ both within and across case analysis. Within cases, we can examine participants' perceptions of group emergent states [93] such as social capital or team morale and ask how these are affected by opportunities or the lack of opportunities for interaction related to stigmergy. We will be especially attentive to gendered differences in perception of these factors. However, it is difficult to identify factors that have deterred some from participating by interviewing only those who do participate. Comparison across settings with different levels of participation may reveal important differences that help explain participation decisions.

Phase II. In phase II, we will determine how the affordances of stigmergic coordination identified in FLOSS and other work can be transferred to a new domain. In choosing a domain, we will identify the tasks and dependencies of different domains (starting from the case studies described above), the kind of work products created and the information that those products might provide to support coordination, cumulating in a detailed analysis of the coordination needs and stigmergic potential of the chosen domain.

Having selected a domain, we will turn to system development. We plan to conserve developer time by building on existing open source tools, e.g., UML editors such as Papyrus + Eclipse or ArgoUML. We have budgeted a small amount of developer time in year 1 to survey existing tools and to assess the work needed to build on them. Starting in year 2 and continuing into year 3, we will build and assess a system that supports the key affordances identified in Phase I. As an example of such a feature, consider how changes are made visible. Developers using single-user drawing tools need to share entire diagrams, which makes it difficult for recipients to identify specific changes. With an online shared drawing tool (e.g., gliffy or LucidChart), changes are visible line by line, so it may be hard to understand the intent of changes. It may be that grouping changes in semantically-meaningful chunks and providing notifications in a form like code check-in messages better supports coordination by making it easier to understand what has changed and why. We expect that prior work on activity displays [20, 21, 55] and translucency [66, 67] will be informative in the design process, as will be a study of the features of tools currently used by the teams (as identified in Phase I). On the other hand, combinability of contributions is also hypothesized to be important, but this feature is not explicitly addressed in prior work. As noted above, making contributions to a diagram combinable will be challenging. It will also be important to develop the social affordances that support stigmergy. Implementing social features may require training for users, provision of other kinds of information (e.g., overviews to help build mental models of the structure of the work) or careful selection of the tasks to be supported. Again, the results of Phase I will provide a set of working hypotheses about these features.

Once the system is developed, we will study people using it in the new domain to test our understanding of the affordance that support stigmergic coordination and to iterate the system design. There are several constraints that guide our selection of study participants. First, we want a team that has a real need to perform a collaborative task that involves a shared computer-based work product. Second, the goal of this phase is not to simply evaluate the system as a black box, but to observe in detail how people use the system and how its affordances affect their ability to coordinate. The need for intensive data collection means we need a team with which we can have in-depth and on-going interaction. Given these constraints, our initial plan is to use the tools with students in online classes working on a group project that requires extensive coordination and collaboration. For UML design, a class in systems analysis, such as the one taught by the PI, would be suitable. However, this decision will be revisited as the system is developed. Student teams are convenient, but have significant limitations for a study of team

coordination. A particular problem for this study is that students may not be sufficiently familiar with the genres of the work to be able to recognize them as cues for coordination. Depending on the domain, it may be possible and desirable to observe non-student groups in a suitably in-depth way.

Nonetheless, our plan to observe student groups creates a natural rhythm for system development. An initial prototype system will be designed and implemented during the fall of year 2 based on the findings from Phase I and a detailed analysis of the tasks, resources and dependencies of the target domain (as described above). During the fall semester, teams can be observed using the unmodified system to augment the domain analysis. Observation of the use of the new system by volunteer teams will be done during the spring of year 2 and findings used for a major iteration of the system during the summer of year 3. Two more rounds of observation with minor iterations will be undertaken in the rest of year 3.

Data about teams' use of the system will be collected via observation, interviews and logs of system use. The observation and interview protocols developed in Phase I will be adapted to more directly address the system features being implemented, and refined as data collection continues. Interviews will be open-ended to allow participants to raise additional issues or concerns. The rich data available from these approaches will serve the goal of assessing whether the implemented features and other socio-technical affordances are effective in enabling stigmergic coordination and to identify possible improvements if not, thus providing an evaluation of the overall project outcomes. Once the system has reached a sufficient level of stability, we plan to release it as an open source project. This release should make it possible to collect data on how teams in general use the software and if they find it effective in supporting coordination, collecting less-rich data on a broader group of users.

Management plan

Based on a preliminary assessment of the effort required, we are requesting funding for one graduate student, a programmer and the PI. The PI will work during the summer on project management, research design and system design, and supervise the graduate student and programmer during the academic year. He will take particular responsibility for selection of mini-case settings and the target domain, overall system design and report writing. The graduate student will support the PI in participant selection, protocol development, domain selection and requirements development. She will have primary responsibility for data collection and analysis, under the oversight of the PI. The programmer will do initial planning and assessment of possible starting systems in year 1 and system development and refinement in years 2 and 3. An initial project activity will be the development of a more detailed timeline against which progress will be measured.

Expected contributions

Intellectual merit. The study has several expected intellectual contributions. First, the empirical study should provide evidence for (or possibly against) stigmergic coordination as a mode of coordination in distributed work. At present, the evidence for stigmergy is mostly indirect. These findings will extend current CSCW research on transparency by providing a theoretical framing for its impact on coordination. As well, the study will identify possible negative outcomes from reliance on stigmergy. Most importantly, the study will identify the socio-technical affordances that enable the use of stigmergy in FLOSS development. Knowing these affordances will provide a basis for designing shared-work systems that support stigmergy. Stigmergic coordination could be useful in a broad range of settings where team members seek to coordinate work while reducing the overhead of explicit communications (indeed, Carroll, et al. [22] suggest that collaboration tools could be made that would be preferable to face-to-face work). The results will also illuminate the possibilities and limits on the transfer of coordination mechanisms from FLOSS development to other domains. The study proposes a plan to study and support stigmergy based on a theoretical model of coordination and the PI's prior research.

Broader impacts of the proposed work. The study has several expected broader impacts. First, implementing a novel mode of coordination could be transformative for the conduct of online work and perhaps collaborative work more generally. Second, if it seems that reliance on stigmergy is off putting for potential group participants (women in particular), then we can study how that happens. Understanding gender differences in the acceptance of this mode of work could help explain and potentially mitigate the huge gap currently observed in participation in FLOSS development in particular

and in online groups more generally. Third, the developed software will be released as open source for use in future research (thus contributing to the infrastructure for research) and potentially for use by distributed workers (thus benefiting society). The identified mechanisms could also be built into other collaborative systems to support stigmergic coordination in other settings. For example, authors [e.g., 3, 5, 22, 29, 105] have noted the potential benefit of stigmergic coordination (or similar processes) for emergency responders, as it could enable better coordination without increasing communication cost. Fourth, the project will provide educational opportunities for a doctoral student and as well for students who will use the system as it is developed. Finally, the PI currently advises one minority PhD student and has had minority undergraduates as REU students on other projects. He will seek opportunities to involve other members of underrepresented groups in the project, e.g., through the Ronald E. McNair Post-Baccalaureate Achievement Program at Syracuse University.

Results from prior NSF funding

The PI for this proposal has been funded by several NSF grants within the past 48 months, though leadership on these projects shifted to replacement PIs while the PI was at NSF. Three of these grants have been on the topic of citizen science. This work has led to 3 journal papers [116, 162, 166], 14 conference proceedings [33, 47, 94, 95, 111, 112, 128-130, 158-160, 164, 165] and numerous conference presentations [e.g., 79, 127, 131, 161, 163].

The grant most relevant to this proposal is IIS-1111107, "SOCS: Socially Intelligent Computing for Coding of Qualitative Data" (current PI Nancy McCracken), Sept 2011–August 2015, $779,831.

Summary of results and accomplishments: The goal of this project (called SOCQA) is to develop and evaluate a research tool to support qualitative social science research, specifically content analysis. The tool applies Natural Language Processing (NLP) and Machine Learning (ML) with an active learning approach that incorporates feedback from human coders. A further goal is to identify the characteristics of content codes that make them easier or harder to automatically identify. The current status of the SOCQA project is that initial system development is complete, and experiments with the tool are ongoing. The project has given the PI useful experience in managing system development.

Intellectual merit: The main focus of the work to date has been system design and implementation and coding the initial sample of data to input to the ML. The researchers have had to refine ML techniques to deal with the kind of data found in content analysis, e.g., down sampling to handle very imbalanced data [80], identifying feature sets that are useful for qualitative data analysis, and experimenting with settings for active learning that balance the preferences of the human analysts and the ML. A major problem has been that some codes occur quite rarely; it is hoped that active learning may be effective in identifying more instances of the code to improve model performance. The work has been presented at a conference [169] and in a journal publication [34] and discussed at workshops [168, 170].

Broader impacts: The SOCQA project is expected to have a broader impact by developing a tool useful for qualitative data analysis, which will be a contribution to the infrastructure for science. It may have an indirect impact by enabling research on a variety of societally important topics. It has had an impact on education by involving several doctoral and (through REU supplements) undergraduate students in the research. These students have learned about NLP and ML techniques, about qualitative data analysis and about FLOSS, the topic of the case used to develop the system. As well, one of the REU students was a member of an underrepresented group, so the project contributed to increasing diversity of the scientific workforce.

References

1 Ahuja, Manju K.; Carley, Kathleen and Galletta, Dennis F.: Individual performance in distributed design groups: An empirical study, in *Proceedings of the SIGMIS Computer Personnel Research Conference*, San Francisco, CA (1997), pp. 160–170

2 Aksulu, Altay and Wade, Michael R.: A comprehensive review and synthesis of open source research, *Journal of the Association for Information Systems*, 2010, 11(11), pp. 576–656

3 Aldunate, Roberto G.; Schmidt, Klaus Nicholas and Herrera, Oriel: Enabling communication in emergency response environments, *Sensors*, 2012, 12(5), pp. 6380–6394, doi: 10.3390/s120506380

4 Alho, Kari and Sulonen, Reijo: Supporting virtual software projects on the Web. Paper presented at the *Workshop on Coordinating Distributed Software Development Projects, 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Palo Alto, CA, USA (17–19 June 1998)

5 Aminoff, Hedvig: *Coordination in Emergency Management from a Joint Cognitive Systems Perspective*. Masters Thesis, Linköping University, Sweden (2007), Available from: http://www.diva-portal.org/smash/get/diva2:402838/FULLTEXT01.pdf

6 Annabi, Hala; Crowston, Kevin and Heckman, Robert: Group learning in the early years of Apache web server, in *Proceedings of the IFIP WG 2.13 Working Conference on Open Source Systems*, Lake Como, Italy (8–9 June 2006), pp. 77–90

7 Arafat, Oliver and Riehle, Dirk: The commit size distribution of open source software, in *Proceedings of the Hawaii International Conference on System Sciences (HICSS-42)*, Hawaii, US (January 2009), pp. 1–8

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

9 Bakhtin, Mikhail Mikhailovich: The problem of speech genres, in Emerson, C., and Holquist, M. (Eds.): *Speech Genres and Other Late Essays: M.M. Bakhtin* (University of Texas Press, 1986), pp. 60–102

10 Beck, Eevi E. and Bellotti, Victoria M. E.: Informed opportunism as strategy: Supporting coordination in distributed collaborative writing, in *Proceedings of the European Conference on Computer-Supported Cooperative Work*, Milan, Italy (13–17 September 1993), pp. 233–248, doi: 10.1007/978-94-011-2094-4_16

11 Bernstein, Ethan S: The transparency paradox: A role for privacy in organizational learning and operational control, *Administrative Science Quarterly*, 2012, 57(2), pp. 181–216

12 Blincoe, Kelly; Valetto, Giuseppe and Goggins, S: Leveraging task contexts for managing developers' coordination, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)* (2012), pp. 1351–1360

13 Blincoe, Kelly; Valetto, Giuseppe and Goggins, Sean: Proximity: A measure to quantify the need for developers' coordination, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, New York, NY, USA (2012), pp. 1351–1360, doi: 10.1145/2145204.2145406

14 Bolici, Francesco; Howison, James and Crowston, Kevin: Stigmergic coordination in FLOSS development teams: Integrating explicit and implicit mechanisms, *Cognitive Systems Research*, In press

15 Bowker, Geoffry C. and Star, Susan Leigh: Knowledge and information in international information management: Problems of classification and coding, in Bud-Frierman, L. (Ed.): *Information*

Acumen: The Understanding and Use of Knowledge in Modern Business (Routledge, 1994), pp. 187–213

16 Bowker, Geoffry C. and Star, Susan Leigh: *Sorting Things Out: Classification and Its Consequences* (MIT Press, 1999)

17 Braunschweig, Brandt and Seaman, Carolyn: An examination of shared understanding in free/libre open source project maintenance. Paper presented at the *International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)* (2013) pp. 113–116

18 Britton, L. C.; Wright, M. and Ball, D. F.: The use of co-ordination theory to improve service quality in executive search, *Service Industries Journal*, 2000, 20(4), pp. 85–102

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

20 Carroll, John M.; Neale, Dennis C.; Isenhour, Philip L.; Rosson, Mary Beth and McCrickard, D.Scott: Notification and awareness: Synchronizing task-oriented collaborative activity, *International Journal of Human-Computer Studies*, 2003, 58(5), pp. 605–632, doi: 10.1016/S1071-5819(03)00024-7

21 Carroll, John M.; Rosson, Mary Beth; Convertino, Gregorio and Ganoe, Craig H.: Awareness and teamwork in computer-supported collaborations, *Interacting with Computers*, 2006, 18(1), pp. 21–46, doi: 10.1016/j.intcom.2005.05.005

22 Carroll, John M.; Rosson, Mary Beth; Farooq, Umer and Xiao, Lu: Beyond being aware, *Information and Organization*, 2009, 19(3), pp. 162–185, doi: 10.1016/j.infoandorg.2009.04.004

23 Cataldo, Marcelo and Herbsleb, James D.: Communication networks in geographically distributed software development, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, San Diego, CA, USA (2008), pp. 579–588

24 Cataldo, Marcelo and Herbsleb, James D.: Coordination breakdowns and their impact on development productivity and software failures, *IEEE Transactions on Software Engineering*, 2013, 39(3), pp. 343–360, doi: 10.1109/TSE.2012.32

25 Christensen, Lars Rune: Practices of stigmergy in architectural work, in *Proceedings of the ACM Conference on Supporting Group Work (Group)*, Sanibel Island, FL (2007)

26 Christensen, Lars Rune: The logic of practices of stigmergy: Representational artifacts in architectural design, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, New York, NY, USA (2008), pp. 559–568

27 Christensen, Lars Rune: Stigmergy in human practice: Coordination in construction work, *Cognitive Systems Research*, 2013, 21, pp. 40–51

28 Christensen, Lars Rune: Practices of stigmergy in the building process, *Computer Supported Cooperative Work (CSCW)*, 2014, 23(1), pp. 1–19, doi: 10.1007/s10606-012-9181-3

29 Convertino, Gregorio; Ganoe, Craig H.; Schafer, Wendy A.; Yost, Beth and Carroll, John M.: A multiple view approach to support common ground in distributed and synchronous geo-collaboration, in *Proceedings of the Third International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV)* (July 2005), pp. 121–132, doi: 10.1109/CMV.2005.2

30 Conway, Melvin E.: How do committees invent, *Datamation*, 1968, 14(4), pp. 28–31

31 Crowston, Kevin: A coordination theory approach to organizational process design, *Organization Science*, 1997, 8(2), pp. 157–175

32 Crowston, Kevin: The bug fixing process in proprietary and free/libre open source software: A coordination theory analysis, in Grover, V., and Markus, M.L. (Eds.): *Business Process Transformation* (M. E. Sharpe, 2008)

33 Crowston, Kevin: Amazon Mechanical Turk: A research tool for organizations and information systems scholars, in *Proceedings of the IFIP Working Group 8.2 Conference: Shaping the Future of ICT Research: Methods and Approaches*, Tampa, FL (December 2012), pp. 210–221

34 Crowston, Kevin; Allen, Eileen E. and Heckman, Robert: Using natural language processing for qualitative data analysis, *International Journal of Social Research Methodology*, 2012, 15(6), pp. 523–543, doi: 10.1080/13645579.2011.625764

35 Crowston, Kevin; Annabi, Hala and Howison, James: Defining open source software project success, in *Proceedings of the International Conference on Information Systems (ICIS)*, Seattle, WA (2003)

36 Crowston, Kevin; Annabi, Hala; Howison, James and Masango, Chengetai: Effective work practices for Software Engineering: Free/Libre Open Source Software Development. Paper presented at the *Workshop on Interdisciplinary Software Engineering Research (WISER), SIGSOFT 2004/FSE-12 Conference*, Newport Beach, CA (2004)

37 Crowston, Kevin; Annabi, Hala; Howison, James and Masango, Chengetai: Towards a portfolio of FLOSS project success measures. Paper presented at the *Workshop on Open Source Software Engineering, 26th International Conference on Software Engineering*, Edinburgh (2004)

38 Crowston, Kevin; Annabi, Hala; Howison, James and Masango, Chengetai: Effective work practices for FLOSS development: A model and propositions, in *Proceedings of the Hawai'i International Conference on System Science (HICSS-38)*, Big Island, Hawai'i (5–9 January 2005)

39 Crowston, Kevin and Howison, James: The social structure of free and open source Software development, *First Monday*, 2005, 10(2)

40 Crowston, Kevin and Howison, James: Hierarchy and centralization in Free and Open Source Software team communications, *Knowledge, Technology and Policy*, 2006, 18(4), pp. 65–85

41 Crowston, Kevin; Howison, James; Masango, Chengetai and Eseryel, U. Yeliz: The role of face-to-face meetings in technology-supported self-organizing distributed teams *IEEE Transactions on Professional Communications*, 2007, 50(3)

42 Crowston, Kevin and Kammerer, Ericka: Communicative style and gender differences in computer-mediated communications, in Ebo, B. (Ed.): *Cyberghetto or Cybertopia: Race, Class and Gender on the Internet* (Praeger, 1998), pp. 185–204

43 Crowston, Kevin and Kammerer, Ericka: Coordination and collective mind in software requirements development, *IBM Systems Journal*, 1998, 37(2), pp. 227–245

44 Crowston, Kevin; Kwaśnik, Barbara H. and Rubleske, Joe: Problems in the use-centered development of a taxonomy of web genres, in Mehler, A., Sharoff, S., and Santini, M. (Eds.): *Genres on the Web: Computational Models and Empirical Studies* (Springer, 2010)

45 Crowston, Kevin; Li, Qing; Wei, Kangning; Eseryel, U. Yeliz and Howison, James: Self-organization of teams for free/libre open source software development, *Information and Software Technology*, 2007, 49(6), pp. 564–575

46 Crowston, Kevin and Osborn, Charlie S.: A coordination theory approach to process description and redesign, in Malone, T.W., Crowston, K., and Herman, G. (Eds.): *Organizing Business Knowledge: The MIT Process Handbook* (MIT Press, 2003)

47 Crowston, Kevin and Prestopnik, Nathan R.: Motivation and data quality in a citizen science game: A design science evaluation, in *Proceedings of the Hawai'i International Conference on System Science (HICSS-46)*, Wailea, HI (2013)

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

49 Crowston, Kevin and Scozzi, Barbara: Bug fixing practices within Free/Libre Open Source Software development teams, *Journal of Database Management*, 2008, 19(2), pp. 1–30, doi: 10.4018/jdm.2008040101

50 Crowston, Kevin; Wei, Kangning; Howison, James and Wiggins, Andrea: Free/libre open source software development: what we know and what we do not know, *ACM Computing Surveys*, 2012, 44(2), pp. 7:1–7:35

51 Crowston, Kevin; Wei, Kangning; Li, Qing; Eseryel, U. Yeliz and Howison, James: Coordination of free/libre open source software development, in *Proceedings of the International Conference on Information Systems (ICIS)*, Las Vegas, NV, USA (2005)

52 Crowston, Kevin and Williams, Marie: Reproduced and emergent genres of communication on the World Wide Web, *Information Society*, 2000, 16(3), pp. 201–215

53 Curtis, Bill; Krasner, Herb and Iscoe, Neil: A field study of the software design process for large systems, *Communications of the ACM*, 1988, 31(11), pp. 1268–1287

54 Curtis, Bill; Walz, Diane and Elam, Joyce J.: Studying the process of software design teams, in *Proceedings of the International Software Process Workshop On Experience With Software Process Models*, Kennebunkport, Maine, United States (October 10-13 1990), pp. 52–53

55 Dabbish, Laura and Kraut, Robert: Research Note—Awareness Displays and Social Motivation for Coordinating Communication, *Information Systems Research*, 2008, 19(2), pp. 221–238, doi: 10.1287/isre.1080.0175

56 Dabbish, Laura and Kraut, Robert E.: Controlling interruptions: Awareness displays and social motivation for coordination, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, New York, NY, USA (2004), pp. 182–191

57 Dabbish, Laura; Stuart, Colleen; Tsay, Jason and Herbsleb, Jim: Social coding in GitHub: Transparency and collaboration in an open software repository, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, New York, NY, USA (2012), pp. 1277–1286, doi: 10.1145/2145204.2145396

58 Dabbish, Laura; Stuart, Colleen; Tsay, Jason and Herbsleb, Jim: Leveraging transparency, *IEEE Software*, 2013, 30(1), pp. 37–43, doi: 10.1109/MS.2012.172

59 Dabbish, Laura; Stuart, H. Colleen; Tsay, Jason and Herbsleb, James D.: Transparency and coordination in peer production, (2014), Available from: http://arxiv.org/abs/1407.0377

60 Dalle, Jean-Michel and David, Paul A: Motivation and coordination in Libre software development: A stygmergic simulation perspective on large community-mode projects. Paper presented at the *DRUID-SCANCOR Conference*, Stanford University (2008), Available from: http://www.researchgate.net/publication/228383396/file/9fcfd50e737e4d2c0f.pdf

61 de Souza, Pedro Sérgio: *Asynchronous Organizations for Multi-Algorithm Problems*. Doctoral Thesis, Carnegie-Mellon University (1993)

62 den Besten, Matthijs; Gaio, Loris; Rossi, Alessandro and Dalle, J-M: Using metadata signals to support stigmergy. Paper presented at the *IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)* (2010) pp. 131–135

63 Dipple, Aiden; Raymond, Kerry and Docherty, Michael: General theory of stigmergy: Modelling stigma semantics, *Cognitive Systems Research*, 2014, 31–32(0), pp. 61–92, doi: 10.1016/j.cogsys.2014.02.002

64 Dourish, Paul and Bellotti, Victoria: Awareness and coordination in shared workspaces, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, Toronto, Ontario, Canada (1992), pp. 107–114, doi: 10.1145/143457.143468

65 Elliot, Mark: Stigmergic collaboration: The evolution of group work, *m/c journal*, 2006, 9(2)

66 Erickson, Thomas: Designing visualizations of social activity: Six claims, in: *Extended Abstracts on Human Factors in Computing Systems* (ACM, 2003), pp. 846–847, doi: 10.1145/765891.766027

67 Erickson, Thomas; Halverson, Christine; Kellogg, Wendy A.; Laff, Mark and Wolf, Tracee: Social translucence: Designing social infrastructures that make collective activity visible, *Communications of the ACM*, 2002, 45(4), pp. 40–44, doi: 10.1145/505248.505270

68 Erickson, Thomas and Kellogg, Wendy A.: Social translucence: an approach to designing systems that support social processes, *ACM Transactions on Computer-Human Interaction*, 2000, 7(1), pp. 59–83, doi: 10.1145/344949.345004

69 Eseryel, U. Yeliz and Eseryel, Deniz: Action-embedded transformational leadership in self-managing global information systems development teams, *Journal of Strategic Information Systems*, 2013, 22, pp. 103–120

70 Espinosa, J. Alberto; Kraut, Robert E.; Lerch, Javier F.; Slaughter, Sandra A.; Herbsleb, James D. and Mockus, Audris: Shared mental models and coordination in large-scale, distributed software development, in *Proceedings of the International Conference on Information Systems (ICIS)*, New Orleans, LA (2001), pp. 513–518

71 Espinosa, J. Alberto; Slaughter, Sandra A.; Kraut, Robert and Herbsleb, James D.: Familiarity, complexity, and team performance in geographically distributed software development, *Organization Science*, 2007, 18(4), pp. 613–630

72 Espinosa, J. Alberto; Slaughter, Sandra A.; Kraut, Robert and Herbsleb, James D.: Team knowledge and coordination in geographically distributed software development, *Journal of Management Information Systems*, 2007, 24(1), pp. 135–169

73 Flores, Fernando; Graves, Michael; Hartfield, Brad and Winograd, Terry: Computer systems and the design of organizational interaction, *ACM Transactions on Office Information Systems*, 1988, 6(2), pp. 153–172

74 Galbraith, Jay R.: *Designing Complex Organizations* (Addison-Wesley, 1973)

75 Gerson, Elihu M. and Star, Susan Leigh: Analyzing due process in the workplace, *ACM Transactions on Office Information Systems*, 1986, 4(3), pp. 257–270

76 Ghosh, Rishab Aiyer: Understanding Free Software Developers: Findings from the FLOSS Study, in Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K. (Eds.): *Making Sense of the Bazaar: Perspectives on Open Source and Free Software* (MIT Press, 2005), pp. 23–45

77 Giddens, Anthony: *The Constitution of Society: Outline of the Theory of Structuration* (University of California, 1984)

78 Grassé, Pierre-Paul: La reconstrution du nid et les coordinations inter-individuelles chez Bellicositermes natalensis et Cubitermes sp. La théorie de la stigmergie: Essai d'interprétation du comportament de termites constructeurs, *Insectes Sociaux*, 1959, 6(1), pp. 41–80, doi: 10.1007/BF02223791

79 Hassman, Katie DeVries; Mugar, Gabriel; Østerlund, Carsten and Jackson, Corey: Learning at the seafloor, looking at the sky: The relationship between individual tasks and collaborative engagement in two citizen science projects (Poster). Paper presented at the *Computer Supported Collaborative Learning Conference* (2013), Available from: http://citsci.syr.edu/sites/crowston.syr.edu/files/Posterv2.pdf

80 He, Haibo and Garcia, Edwardo A.: Learning learning from imbalanced data, *IEEE Transactions on Knowledge and Data Engineering*, 2009, 21(9), pp. 1263–1284, doi: 10.1109/TKDE.2008.239

81 Heckman, Robert; Crowston, Kevin; Eseryel, U. Yeliz; Howison, James; Allen, Eileen and Li, Qing: Emergent decision-making practices In Free/Libre Open Source Software (FLOSS) development

teams, in *Proceedings of the IFIP WG 2.13 Working Conference on Open Source Systems*, Limerick, Ireland (11–15 June 2007)

82 Heckman, Robert; Crowston, Kevin; Li, Qing; Allen, Eileen E.; Eseryel, Yeliz; Howison, James and Wei, Kangning: Emergent decision-making practices in technology-supported self-organizing distributed teams, in *Proceedings of the International Conference on Information Systems (ICIS)*, Milwaukee, WI, 10–13 Dec (2006)

83 Heckman, Robert; Crowston, Kevin and Misiolek, Nora: A structurational perspective on leadership in virtual teams, in *Proceedings of the IFIP Working Group 8.2/9.5 Working Conference on Virtuality and Virtualization*, Portland, OR (2007), pp. 151–168

84 Herbsleb, James D. and Grinter, Rebecca E.: Splitting the organization and integrating the code: Conway's law revisited, in *Proceedings of the International Conference on Software Engineering (ICSE)*, Los Angeles, CA (1999), pp. 85–95

85 Herbsleb, James D.; Mockus, Audris; Finholt, Thomas A. and Grinter, Rebecca E.: An empirical study of global software development: Distance and speed, in *Proceedings of the International Conference on Software Engineering (ICSE)*, Toronto, Canada (2001), pp. 81–90

86 Heylighen, Francis: Why is open access development so successful? Stigmergic organization and the economics of information, in Lutterbeck, B., Bärwolff, M., and Gehring, R.A. (Eds.): *Open Source Jahrbuch 2007* (Lehmanns Media, 2007)

87 Hill, William C.; Hollan, James D.; Wroblewski, Dave and McCandless, Tim: Edit wear and read wear, in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, New York, NY, USA (1992), pp. 3–9

88 Hollan, Jim and Stornetta, Scott: Beyond being there, in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)* (1992)

89 Howison, James: *Alone Together: A Socio-Technical Theory of Motivation, Coordination and Collaboration Technologies in Organizing for Free and Open Source Software Development*. Doctoral Dissertation, Syracuse University (2009)

90 Howison, James and Crowston, Kevin: The perils and pitfalls of mining SourceForge. Paper presented at the *Workshop on Mining Software Repositories, 26th International Conference on Software Engineering*, Edinburgh, Scotland (2004)

91 Howison, James and Crowston, Kevin: Collaboration through open superposition: A theory of the open source way, *MIS Quarterly*, 2014, 38(1), pp. 29–50

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

93 Ilgen, Daniel R.; Hollenbeck, John R.; Johnson, Michael and Jundt, Dustin: Teams in organizations: From input-process-output models to IMOI models, *Annual Review of Psychology*, 2005, 56(1), pp. 517–543, doi: 10.1146/annurev.psych.56.091103.070250

94 Jackson, Corey Brian; Østerlund, Carsten; Mugar, Gabriel; Crowston, Kevin and Hassman, Katie DeVries: Motivations for sustained participation in crowdsourcing: The role of talk in a citizen science case study, in *Proceedings of the Hawai'i International Conference on System Science (HICSS-48)*, Koloa, HI (January 2015)

95 Jackson, Corey; Østerlund, Carsten; Maidel, Veronica; Mugar, Gabriel and Crowston, Kevin: Which way did they go? Newcomer movement through the Zooniverse, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW 2016)*, San Francisco, CA (27 Feb–2 Mar 2106)

96 Jacob, Elin K: The everyday world of work: Two approaches to the investigation of classification in context, *Journal of Documentation*, 2001, 57(1), pp. 76–99

97 Kalliamvakou, Eirini; Damian, Daniela; Singer, Leif and German, Daniel M: The code-centric collaboration perspective: Evidence from github, (2014), Technical Report DCS-352-IR, University of Victoria. Available from: http://thesegalgroup.org/wp-content/uploads/2014/04/code-centric.pdf

98 Ke, Weiling and Zhang, Ping: Motivations in Open Source Software Communities: The Mediating Role of Effort Intensity and Goal Commitment, *International Journal of Electronic Commerce*, 2009, 13(4), pp. 39–66

99 Kittur, Aniket and Kraut, Robert E: Harnessing the wisdom of crowds in Wikipedia: Quality through coordination, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)* (2008), pp. 37–46

100 Latour, Bruno: Visualisation and cognition: Drawing things together, in Lynch, M., and Woolgar, S. (Eds.): *Representation in Scientific Practice* (MIT Press, 1990)

101 Lawrence, Paul R. and Lorsch, Jay W.: *Organization and Environment* (Harvard Business School, 1967)

102 Li, Qing; Heckman, Robert; Allen, Eileen; Crowston, Kevin; Eseryel, U. Yeliz; Howison, James and Wiggins, Andrea: Asynchronous decision-making in distributed teams (Poster). Paper presented at the *ACM Conference on Computer Supported Cooperative Work (CSCW)* San Diego, CA (8–12 November 2008)

103 Li, Qing; Heckman, Robert; Crowston, Kevin; Howison, James; Allen, Eileen E. and Eseryel, U. Yeliz: Decision-making paths in technology-supported self-organizing distributed teams, in *Proceedings of the International Conference on Information Systems (ICIS)*, Paris, France (14-17 December 2008)

104 Malone, Thomas W. and Crowston, Kevin: The interdisciplinary study of coordination, *Computing Surveys*, 1994, 26(1), pp. 87–119

105 Marsden, Janet: Determining the role of geospatial technologies for stigmergic coordination in situation management: Implications of the wireless grid, in *Proceedings of the IEEE First International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)* (22-24 Feb 2011), pp. 131–135, doi: 10.1109/COGSIMA.2011.5753431

106 Martins, Luis L.; Gilson, Lucy L. and Maynard, M. Travis: Virtual teams: What do we know and where do we go from here?, *Journal of Management*, 2004, 30(6), pp. 805–835, doi: 10.1016/j.jm.2004.05.002

107 Miles, Matthew B. and Huberman, A. M.: *Qualitative Data Analysis: An Expanded Sourcebook* (Sage Publications, 2nd edn, 1994)

108 Mintzberg, Henry: *The Structuring of Organizations* (Prentice-Hall, 1979)

109 Misiolek, Nora and Heckman, Robert: Patterns of emergent leadership in virtual teams, in *Proceedings of the Hawai'i International Conference on System Science (HICSS-38)*, Big Island, HI (2005)

110 Mockus, Audris; Fielding, Roy T. and Herbsleb, James D.: A case study of Open Source Software development: The Apache server, in *Proceedings of the International Conference on Software Engineering (ICSE)* (2000), pp. 11 pages

111 Mugar, Gabriel; Østerlund, Carsten; Hassman, Katie DeVries; Crowston, Kevin and Jackson, Corey Brian: Planet Hunters and Seafloor Explorers: Legitimate peripheral participation through practice proxies in online citizen science, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)* (February 2014)

112 Mugar, Gabriel; Østerlund, Carsten; Jackson, Corey and Crowston, Kevin: Being present in online communities: Learning in citizen science, in *Proceedings of the Communities and Technologies (C&T) Conference*, Limerick, Ireland (27–30 June 2015)

113 Musil, Juergen; Musil, Angelika and Biffl, Stefan: Towards a coordination-centric architecture metamodel for social web applications, in *Proceedings of the European Conference on Software Architecture (ECSA 2014)*, Vienna, Austria (August 25–29 2014), pp. 106–113

114 Nafus, Dawn; Leach, James and Krieger, Bernhard: Gender: Integrated Report of Findings, (2006), UCAM, University of Cambridge. Available from: http://flosspols.org/deliverables/FLOSSPOLS-D16-Gender_Integrated_Report_of_Findings.pdf

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

116 Newman, Greg; Wiggins, Andrea; Crall, Alycia; Graham, Eric; Newman, Sarah and Crowston, Kevin: The future of citizen science: Emerging technologies and shifting paradigms, *Frontiers in Ecology and the Environment*, 2012, 10, pp. 298–304

117 O'Leary, Michael Boyer; Orlikowski, Wanda J. and Yates, JoAnne: Distributed work over the centuries: Trust and control in the Hudson's Bay Company, 1670–1826, in Hinds, P., and Kiesler, S. (Eds.): *Distributed Work* (MIT Press, 2002), pp. 27–54

118 Ocker, Rosaliel J. and Fjermestad, Jerry: High versus low performing virtual design teams: A preliminary analysis of communication, in *Proceedings of the Hawai'i International Conference on System Sciences (HICSS-33)* (2000), pp. 10 pages

119 Orlikowski, Wanda J. and Yates, JoAnne: Genre repertoire: The structuring of communicative practices in organizations, *Administrative Science Quarterly*, 1994, 33, pp. 541–574

120 Ortega, Felipe: *Wikipedia: A Quantitative Analysis*. Doctoral dissertation, Universidad Rey Juan Carlos (2009)

121 Østerlie, Thomas and Jaccheri, Letizia: A critical review of software engineering research on open source software development. Paper presented at the *AIS SIGSAND European Symposium on Systems Analysis and Design*, Gdansk, Poland (5 June 2007)

122 Østerlund, Carsten: Genre Combinations: A Window into Dynamic Communication Practices, *Journal of Management Information Systems*, 2007, 23(4), pp. 81–108

123 Østerlund, Carsten: The materiality of communicative practice: The boundaries and objects of an emergency room genre, *Scandinavian Journal of Information Systems*, 2008, 20(1), pp. 7–40

124 Østerlund, Carsten; Sawyer, Steve and Kazianus, Elizabeth: Documenting Work: A Methodological Window into Coordination in Action, in *Proceedings of the Conference of the European Group for Organizational Studies (EGOS)* (2010)

125 Parunak, H. V.: A survey of environments and mechanisms for human-human stigmergy, in Weyns, D., Parunak, H.V.D., and Michel, F. (Eds.): *Environments for Multi-Agent Systems II* (2006), pp. 163–186, doi: 10.1007/11678809_10

126 Pfeffer, Jeffery and Salancik, G. R.: *The External Control of Organizations: A Resource Dependency Perspective* (Harper & Row, 1978)

127 Prestopnik, Nathan and Crowston, Kevin: Purposeful gaming & socio-computational systems: A citizen science design case. Paper presented at the *ACM Conference on Supporting Group Work (Group)*, Sanibel Island, FL, USA (2012)

128 Prestopnik, Nathan and Crowston, Kevin: Purposeful gaming and socio-computational systems: A citizen science design case, in *Proceedings of the ACM Conference on Supporting Group Work (Group)*, Sanibel Island, FL, USA (27--31 October 2012)

129 Prestopnik, Nathan R. and Crowston, Kevin: Citizen science system assemblages: Understanding the technologies that support crowdsourced science, in *Proceedings of the iConference*, Toronto, Ontario (7–10 February 2012)

130 Prestopnik, Nathan R.; Crowston, Kevin and Wang, Jun: Exploring data quality in games with a purpose, in *Proceedings of the iConference*, Berlin, Germany (4–7 March 2014)

131 Prestopnik, Nathan and Souid, Dania: Forgotten island: A story-driven citizen science adventure, in: *CHI Extended Abstracts on Human Factors in Computing Systems* (ACM Press, 2013), pp. 2643–2646

132 Raymond, Eric S.: The cathedral and the bazaar, *First Monday*, 1998, 3(3)

133 Ricci, Alessandro; Viroli, Andrea Omiciniand Mirko; Gardelli, Luca and Oliva, Enrico: Cognitive stigmergy: Towards a framework based on agents and artifacts, in Weyns, D., Parunak, H.V.D., and Michel, F. (Eds.): *Environments for Multi-Agent Systems III* (Springer, 2007), pp. 124–140, doi: 10.1007/978-3-540-71103-2_7

134 Rico, Ramón; Sánchez-Manzanares, Miriam; Gil, Francisco and Gibson, Cristina: Team implicit coordination processes: A team knowledge–based approach, *Academy of Management Review*, 2008, 33(1), pp. 163–184

135 Rossi, Maria Alessandra: Decoding the free/open source puzzle: A survey of theoretical and empirical contributions, in Bitzer, J., and Schroder, P. (Eds.): *The Economics of Open Source Software Development: Analyzing Motivation, Organization, Innovation and Competition in the Open Source Software Revolution* (Elsevier Press, 2004), pp. 15–55

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

137 Scacchi, Walt: The software infrastructure for a distributed software factory, *Software Engineering Journal*, 1991, 6(5), pp. 355–369

138 Scialdone, Michael J.; Heckman, Robert and Crowston, Kevin: Group maintenance behaviours of core and peripheral members of free/libre open source software teams, in *Proceedings of the IFIP WG 2.13 Working Conference on Open Source Systems*, Skövde, Sweden (3–6 June 2009)

139 Scialdone, Michael J.; Li, Na (Lina); Howison, James; Heckman, Robert and Crowston, Kevin: Group maintenance in technology-supported distributed teams, in: *Best Paper Proceedings, Academy of Management Annual Meeting* (2008)

140 Seaman, Carolyn B. and Basili, Victor R.: Communication and organization in software development: An empirical study, *IBM Systems Journal* 1997, 36(4), pp. 550–563, doi: 10.1147/sj.364.0550

141 Secretan, Jimmy: Stigmergic dimensions of online creative interaction, *Cognitive Systems Research*, 2013, 21, pp. 65–74

142 Smith, Dorothy E.: *Institutional Ethnography: A Sociology for People* (AltaMira Press, 2005)

143 Squire, Megan and Crowston, Kevin: FLOSShub, (2015), Available from: http://flosshub.org/

144 Strauss, Anselm: Work and the division of labor, *The Sociological Quarterly*, 1985, 26(1), pp. 1–19

145 Stuart, H Colleen; Dabbish, Laura; Kiesler, Sara; Kinnaird, Peter and Kang, Ruogu: Social transparency in networked information exchange: A theoretical framework, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)* (2012), pp. 451–460

146 Suchman, Lucy A.: Technologies of accountability: Of lizards and aeroplanes, in Button, G. (Ed.): *Technology in Working Order* (Routledge, 1993)

147 Suchman, Lucy A.: Making work visible, *Communications of the ACM*, 1995, 38(9), pp. 56–65

148 Susi, Tarja and Ziemke, Tom: Social cognition, artefacts, and stigmergy: A comparative analysis of theoretical frameworks for the understanding of artefact-mediated collaborative activity, *Cognitive Systems Research*, 2001, 2(4), pp. 273–290

149 Thompson, James D.: *Organizations in Action: Social Science Bases of Administrative Theory* (McGraw-Hill, 1967)

150 Tummolini, Luca and Castelfranchi, Cristiano: Trace signals: The meanings of stigmergy, in Weyns, D., Parunak, H.V.D., and Michel, F. (Eds.): *Environments for multi-agent systems III* (Springer, 2007), pp. 141–156, doi: 10.1007/978-3-540-71103-2_8

151 van Fenema, Paul C.: *Coordination and control of globally distributed software projects*. Doctoral Dissertation, Erasmus University (2002)

152 Viégas, Fernanda B.; Wattenberg, Martin; Kriss, J. and van Ham, F.: Talk before you type: Coordination in Wikipedia, in *Proceedings of the Hawai'i International Conference on System Sciences (HICSS-40)* (2007)

153 Walz, Diane B.; Elam, Joyce J. and Curtis, Bill: Inside a software design team: Knowledge acquisition, sharing, and integration, *Communications of the ACM*, 1993, 36(10), pp. 63–77

154 Watson-Manheim, Mary Beth; Chudoba, Katherine M. and Crowston, Kevin: Perceived discontinuities and constructed continuities in virtual work, *Information Systems Journal*, 2012, 22(1), pp. 29–52, doi: 10.1111/j.1365-2575.2011.00371.x

155 Wayner, Peter: *Free For All* (HarperCollins, 2000)

156 Wei, Kangning; Crowston, Kevin; Li, Na Lina and Heckman, Robert: Understanding group maintenance behavior in Free/Libre Open-Source Software projects: The case of Fire and Gaim, *Information and Management*, 2014, 51(3), pp. 297–309

157 Wexelblat, Alan and Maes, Pattie: Footprints: History-rich tools for information foraging, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA (1999), pp. 270–277

158 Wiggins, Andrea: Organizing from the middle out: Citizen science in the national parks, in *Proceedings of the iConference*, Urbana-Champaign, IL, USA (3–6 February 2010)

159 Wiggins, Andrea: Technology and work practices in citizen science, in *Proceedings of the American Society for Information Science and Technology Annual Meeting*, Pittsburgh, PA (October 2010)

160 Wiggins, Andrea: eBirding: Technology adoption and the transformation of leisure into science, in *Proceedings of the iConference*, Seattle, WA (8–11 February 2011)

161 Wiggins, Andrea and Crowston, Kevin: Designing virtual organizations for citizen science. Paper presented at the *IFIP Working Group 8.2 OASIS Workshop*, Phoenix, AZ (December 2009), Available from: http://citsci.syr.edu/sites/crowston.syr.edu/files/WigginsOASIS2009.pdf

162 Wiggins, Andrea and Crowston, Kevin: Developing a conceptual model of virtual organizations for citizen science, *International Journal of Organizational Design and Engineering*, 2010, 1(1/2), pp. 148–162, doi: 10.1504/IJODE.2010.035191

163 Wiggins, Andrea and Crowston, Kevin: Distributed scientific collaboration: Research opportunities in citizen science. Paper presented at the *Workshop on The Changing Dynamics of Scientific Collaboration, CSCW 2010*, Savannah, GA (February 2010), Available from: http://citsci.syr.edu/sites/crowston.syr.edu/files/WigginsCSCWworkshop_0.pdf

164 Wiggins, Andrea and Crowston, Kevin: From conservation to crowdsourcing: A typology of citizen science, in *Proceedings of the Hawai'i International Conference on System Science (HICSS-44)*, Koloa, HI (January 2011)

165 Wiggins, Andrea and Crowston, Kevin: Goals and tasks: Two typologies of citizen science projects, in *Proceedings of the Hawai'i International Conference on System Science (HICSS-45)*, Wailea, HI (3–7 January 2012)

166 Wiggins, Andrea and Crowston, Kevin: Surveying the citizen science landscape, *First Monday*, 2015, 26(1), doi: 10.5210/fm.v20i1.5520

167 Winograd, Terry: A language/action perspective on the design of cooperative work, *Human Computer Interaction*, 1987, 3, pp. 3–30

168 Yan, Jasy Liew Suet; McCracken, Nancy and Crowston, Kevin: Design of an active learning system with human correction for content analysis. Paper presented at the *Workshop on Interactive Language Learning, Visualization, and Interfaces, 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, MD (June 2014), Available from: http://socqa.org/sites/crowston.syr.edu/files/ILLWorkshop.ACLFormat.04.28.14.final_.pdf

169 Yan, Jasy Liew Suet; McCracken, Nancy and Crowston, Kevin: Semi-automatic content analysis of qualitative data, in *Proceedings of the iConference*, Berlin, Germany (4–7 March 2014)

170 Yan, Jasy Liew Suet; McCracken, Nancy; Zhou, Shichun and Crowston, Kevin: Optimizing features in active machine learning for complex qualitative content analysis. Paper presented at the *Workshop on Language Technologies and Computational Social Science, 52nd Annual Meeting of the Association for Computational Linguistics* Baltimore, MD (June 2014), Available from: http://socqa.org/sites/crowston.syr.edu/files/9_Paper.pdf

171 Yates, Joanne and Orlikowski, Wanda J.: Genres of organizational communication: A structurational approach to studying communication and media, *Academy of Management Review*, 1992, 17(2), pp. 299–327

172 Zacklad, Manuel: Documentarisation processes in documents for action (DofA): The status of annotations and associated cooperation technologies, *Computer Supported Cooperative Work (CSCW)*, 2006, 15(2-3), pp. 205–228

173 Zhang, Wei; Zhao, Haiyan; Jiang, Yi and Jin, Zhi: Stigmergy-Based Construction of Internetware Artifacts, *IEEE Software*, 2015, 32(1), pp. 58–66, doi: 10.1109/ms.2014.133