Organization Science
Vol. 18, No. 4, July–August 2007, pp. 613–630
ISSN 1047-7039 | EISSN 1526-5455 | 07 | 1804 | 0613
DOI 10.1287/orsc.1070.0297
© 2007 INFORMS

Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development

J. Alberto Espinosa
Kogod School of Business, American University, Washington, D.C. 20016, alberto@american.edu
Sandra A. Slaughter
College of Management, Georgia Tech University, Atlanta, Georgia 30332, sandra.slaughter@mgt.gatech.edu
Robert E. Kraut, James D. Herbsleb
Carnegie Mellon University, Pittsburgh, Pennsylvania 15213,
{robert.kraut@cmu.edu, jdh@cs.cmu.edu}

While prior research has found that familiarity is beneficial to team performance, it is not clear whether different kinds of familiarity are more or less beneficial when the work has different types of complexity. In this paper, we theorize how task and team familiarity interact with task and team coordination complexity to influence team performance. We posit that task familiarity is more beneficial with more complex tasks (i.e., tasks that are larger or with more complex structures) and that team familiarity is more beneficial when team coordination is more difficult (i.e., for larger or geographically dispersed teams). Finally, we propose that the effects of task familiarity and team familiarity on team performance are complementary. Based on a field study of geographically distributed software teams, two of our hypotheses are disconfirmed: Our results show that the beneficial effects of task familiarity decline when tasks are more structurally complex and are independent of task size. Conversely, the hypotheses for team familiarity are confirmed as the benefit of team familiarity for team performance is enhanced when team coordination is more challenging—i.e., when teams are larger or geographically dispersed. Finally, surprisingly, we find that task and team familiarity are more substitutive than complementary in their joint effects on team performance: Task familiarity improves team performance more strongly when team familiarity is weak and vice versa. Our study contributes by revealing how different types of familiarity can enhance team performance in a real-world setting where the task and its coordination can be highly complex.

Key words: familiarity; team familiarity; task familiarity; teams; software development; global software development; geographically distributed teams; global software teams; shared work knowledge; team cognition

Introduction

The importance of teams as fundamental units of work in organizations (Hackman 1987, Sproull and Kiesler 1991) has been well-documented (Cohen and Bailey 1997, Harrison et al. 2003, McGrath 1991). Teams are particularly useful when tasks are too large or too complex for a single individual to undertake. However, teamwork is not easy, requiring a substantial amount of coordination among team members, particularly when task activities are interdependent (Thompson 1967, Van de Ven et al. 1976). To be effective, team members need to coordinate and carry out their "taskwork" and "teamwork" activities competently (Klimoski and Mohammed 1994).

When tasks are simple and well-defined, it is easy to identify and understand which parts of the task affect other parts. Performing taskwork activities, however, becomes harder as the number and relatedness of task activities increases because this added "task complexity" makes it more difficult for individuals to integrate the various parts of a task and identify how task components affect each other. Individuals working on these more complex tasks also need to process more information cues (Wood 1986) in order to identify, understand, prioritize, and resolve task component dependency issues, diverting their attention from other task responsibilities and making it more difficult to perform the task.

In addition, when a task is carried out collaboratively by more than one individual, the task activities of the various individuals also need to be coordinated and integrated, further increasing the complexity of the teamwork effort. For example, a task can have a certain inherent level of complexity due to its size and structure. However, this same task can become more or less complex depending on how many people work on it and on how these people are configured geographically because the individuals need to coordinate and integrate their respective work. The challenges of working as a team increase under conditions where it is difficult for members to communicate and coordinate with each other and effectively manage their mutual dependencies (Malone and Crowston 1994). We refer to these more difficult conditions as "team coordination complexity." Specifically, we consider the challenges of team size and dispersion. Larger teams have more productive resources. However, Brooks' concept of the "mythical man month" is based on the fact that adding more team members to a project doesn't necessarily help it

to finish sooner (Brooks 1995). Instead, adding individuals to a team exponentially increases the number of possible dependency links among team members, bringing about substantial coordination and project management overhead. Similarly, when team members are separated geographically, their ability to communicate is hampered through many routes. Among other problems, geographic dispersion makes it challenging for individual members to get acquainted with their colleagues' work skills and habits, identify and access expertise when needed, develop task, presence, and contextual awareness, and manage their respective task dependencies (Herbsleb and Grinter 1999).

We contend that familiarity can help teams handle complexity more effectively. Familiarity helps team members perform their individual task activities and communicate and coordinate their work with their teammates. The concept of familiarity in organizational teams has been defined as "the knowledge that members of a team have about the unique aspects of their work" (Goodman and Garber 1988), such as knowledge about the task itself and about other members on the team (Littlepage et al. 1997). As members of a team work together over time, they become familiar with the task domain and with each other (Katz 1982), and they develop a common knowledge base through which team interaction and location of expert sources in the team can occur (Alavi and Leidner 2001). Studies have shown the positive benefits of familiarity on team performance in mining (Goodman and Leyden 1991), flight simulation (Kanki and Foushee 1989), problem solving (Gruenfeld et al. 1996, Littlepage et al. 1997), and various other tasks (Harrison et al. 2003). However, while it may seem intuitively obvious that familiarity with the task and with team members improves team performance, some of the empirical evidence is not so clear.

Summarizing the mixed results in the familiarity literature, Harrison and colleagues (Harrison et al. 2003) have conjectured that the effects of familiarity on team performance may differ depending on the nature of the task itself. Similarly, Argote and colleagues (Argote et al. 1995, Reagans et al. 2005) have empirically investigated team performance and task complexity in experimental and field settings and have also concluded that the task matters when studying the effects of familiarity. Their experimental study of student teams performing origami tasks found that team performance increased with task experience and decreased with task complexity and that performance on simpler tasks improved more strongly with task experience than on more complex tasks. They also found that team member turnover had a negative effect on performance but this effect was weaker with more complex tasks, possibly because teams with higher turnover were more innovative (Argote et al. 1995). A more recent field study discovered that team performance in medical procedures actually decreased with small increments of task experience in the team but increased substantially once task experience exceeded a certain threshold, perhaps because team members tackled more complex tasks as they gained experience (Reagans et al. 2005). The results of these studies suggest that aspects of teams and task complexity may interact in complicated ways, and that further research is necessary to understand how these interactions may affect team performance. As Simon's (1955) concept of "bounded rationality" implies, individuals have limitations on their computational ability to process all the constraints and feasible choices to solve a complex problem. Furthermore, the solution to a complex problem often involves adapting solutions to similar complex problems solved in the past (Simon 1996). Thus, more familiar team members may be in a better position to cope with complexity.

To the best of our knowledge, no prior study has theorized and empirically investigated how different types of familiarity—task and team—interact with different dimensions of complexity—task and team coordination complexity—in their effects on team performance. Learning more about these interactions is important to better understand how team members deal with the different complexities that increasingly characterize their work environments. In this paper, we theorize how task and team familiarity and complexity complement or offset each other in affecting team performance. Specifically, we investigate whether task familiarity is particularly beneficial when team tasks are more complex, i.e., larger or more structurally complex, and whether team familiarity is more beneficial when team coordination is more difficult, i.e., when teams are larger or geographically dispersed. Finally, we draw on the literature on transactive memory to propose an interaction between task familiarity and team familiarity in their effects on team performance. Brandon and Hollingshead (2004) have suggested that performance is jointly influenced by familiarity with the task, the expertise required to complete the task, and the people who possess this expertise because this familiarity helps locate and access the necessary expertise in the team. However, it is not clear whether familiarity with the task and team are complementary or substitutive, or if they are simply additive in their effects on team performance. In sum, our research addresses this question: *How do task familiarity, team familiarity, and complexity complement or offset each other in affecting team performance*?

Software Product Development

We study teams in a field setting that is often characterized by high levels of complexity: geographically distributed software product development. Software product development is a context well-suited to investigate how

familiarity influences the effects of complexity on performance. It involves the addition and modification of features in a product base by several developers working in teams, who need to manage the tightly coupled interdependencies with each other and the complexities of their various task activities. Software tasks are inherently complex, but this complexity varies greatly depending on characteristics of the software task itself like size and structure, and on the coordination challenges imposed on the team by factors like team size and geographic dispersion, thus making software product development an ideal task for our study.

Theoretical Foundations and Hypotheses

Our premise in this study is that familiarity helps team members cope with complexity. Workers who are familiar with the task and its context are thought to have larger bodies of knowledge, better organization of this knowledge, and better internal representation of problems (Goodman and Shah 1992), all of which enable them to respond to work stimuli more quickly and automatically. A person's familiarity with a task reduces the subjective complexity of the task as experienced by that person (Campbell 1988). Similarly, the coordination complexity experienced by team members when working with other members is reduced when they are familiar with each other. To understand how familiarity offsets some of the negative effects of complexity, it is important to consider how different kinds of familiarity interact with different dimensions of complexity. Familiarity researchers distinguish between task familiarity and team familiarity, arguing that familiarity with the task is important for individual task performance while familiarity with other team members is important for effective team interaction (Harrison et al. 2003, Littlepage et al. 1997). In the following sections, we examine these two types of familiarity and their respective interactions with task complexity and team coordination complexity.

How Task Familiarity Offsets Task Complexity for Team Performance

Goodman and Leyden (1991) reasoned that task familiarity has a positive effect on task performance because every task requires unique configurations of machinery, physical environment, and work activities; therefore, team members' specific knowledge about these aspects of their work can make them more productive. These beneficial effects have been confirmed in empirical studies with mining teams (Goodman and Leyden 1991), experimental problem-solving tasks (Littlepage et al. 1997), medical teams (Reagans et al. 2005), and in software development (Banker and Slaughter 2000, Boehm 1981, Brooks 1995, Curtis et al. 1988, Walz et al. 1993). While the general benefits of task familiarity are apparent from this empirical evidence, it is not clear whether task familiarity can complement or offset different task complexity conditions. In their studies with mining teams, Goodman and Leyden (1991) found that familiarity with different aspects of the task had differential effects on performance, suggesting that task familiarity may complement certain task factors but not others. Goodman and Shah (1992) reasoned that prior familiarity with the task can have a stronger effect on performance when the inherent characteristics of the task make it more difficult for team members to understand it. However, the results from the origami experiment by Argote et al. (1995) suggest the opposite, indicating that familiarity with the task is most beneficial for simpler (not more complex) tasks.

In software development, the size of the task is one characteristic that can make the task complex (Banker et al. 1998, Curtis et al. 1979, Kemerer 1995). Software task size is a function of the amount of work, i.e., the number of new and changed software instructions implemented. These instructions create new functionality or modify the existing functionality of the product base. Implementing the instructions also requires knowledge of which parts of the software product will be affected by the changes. Thus, software developers not only need to understand several functional aspects of the software product base but also the technical details of the associated software code (e.g., file names, variable names, and interface requirements). When a particular software project requires more numerous changes, implementing the various changes becomes more difficult because there are more components that need to be updated and more information, such as variable names, software programs, and interface requirements that need to be taken into account, which in turn increase the knowledge and skill requirements to complete the task (Wood 1986). Thus, all else equal, a software project that requires more instructions to change the functionality of the existing product base will be more complex than one that requires fewer instructions.

Task familiarity is particularly important for productivity in larger software tasks because with a greater number of changes, more of the software product code base is affected. The developers' specific knowledge of the application domain and product base helps them to pinpoint more swiftly and accurately where in the software code base the changes need to be made (Banker and Slaughter 2000). In addition, familiarity with the software and technical environment helps developers to be more efficient in testing and implementing larger changes because they know what needs to be tested and how to implement the changes (Curtis et al. 1988). Thus, we posit:

HYPOTHESIS 1. *Task familiarity and task size interact positively in their effect on team performance, such that the positive effect of task familiarity on team performance is stronger with larger tasks.*

Task complexity increases not only with task size but also with the interrelationships between task activities and with the amount of information that needs to be processed to carry out these activities. This type of complexity is consistent with concepts like task component and coordinative complexity (Wood 1986) and structural complexity (Darcy et al. 2005, Xia and Lee 2005) discussed in the literature. In software development, structural complexity describes the characteristics or attributes of the software artifact that is being created or modified. Generally, software structures composed of more interrelated modules are considered more complex (Darcy et al. 2005). This is because more information needs to be processed during task performance: In addition to comprehending each of the individual software modules, developers must also expend additional cognitive effort to understand how the modules are connected. For example, when changing a module that is related to many other modules, developers must envision how the changes made will affect these other modules. Additional testing may also be required to ensure that the module changes do not negatively affect the performance of related modules. Thus, as the interrelationships among various parts of the software increase, the task becomes more complex and developers need to spend even more time discerning and verifying how their work in one part of the software will affect other parts (Banker et al. 1993, Banker et al. 1998). Indeed, best practices in software development recommend that software architects limit the interdependence between software modules precisely to reduce this type of structural complexity (Darcy et al. 2005).

Although the prior research on software engineering suggests that as software becomes extremely complex, the inherent complexity eventually is irreducible and can not be overcome with further developer experience (Brooks 1995), we would generally expect that greater familiarity should be helpful for completing more complex software tasks. Developers who are more familiar with the functional and structural aspects of the software should be able to integrate the changed modules into the software code base more efficiently (Curtis et al. 1979). In other words, we expect that familiarity with the task will generally be more beneficial when working on more structurally complex tasks. Team members having more familiarity with the task may be better able to anticipate the impacts of their activities on other parts of the task, reducing the effort needed to verify the changes and to resolve problems. Thus, we posit:

HYPOTHESIS 2. *Task familiarity and structural complexity interact positively in their effect on team performance, such that the positive effect of task familiarity on team performance is stronger when the structure of the task is more complex.*

How Team Familiarity Offsets Team Coordination Complexity for Team Performance

We have discussed how greater task size and structural complexity can increase the difficulty of completing a software task for an individual developer. However, when a software task is completed by many developers, it is also necessary to coordinate their efforts. The complexity of coordinating the task will be affected by how easy or difficult it is for the multiple collaborators involved to integrate their various pieces successfully. In addition, the task will become increasingly more difficult when the conditions in which these members operate make it more difficult for them to communicate and exchange information. Team familiarity can help offset some of these problems. The beneficial effects of team familiarity have been found in flight simulation (Kanki and Foushee 1989), medical (Reagans et al. 2005), software (Crowston and Kammerer 1998, Faraj and Sproull 2000), and various experimental tasks (Gruenfeld et al. 1996, Littlepage et al. 1997), and in transactive memory research (Lewis 2003, Liang et al. 1995, Wegner 1995). Teams that have prior experience working together on the same projects start a new task with better expectations about each other, communicate more effectively because they have more common ground and can refer to the same technical terminology (Cramton 2001), are more effective at locating specialized knowledge and coordinating expertise within the team (Faraj and Sproull 2000), have a better understanding of how their individual work contributes to each other's tasks, and better understand how to obtain cooperation from other members.

Because team familiarity helps members identify and access expertise more effectively, we anticipate that when the work environment affects the team's ability to identify and access expertise, it will also influence how strongly team familiarity affects team performance. It is important to note that when team members are familiar with the task and understand the various roles people need to undertake to carry out the task (e.g., developer, testing engineer, software architect), this familiarity with roles can also help members access expertise. As Brandon and Hollingshead (2004) have noted, it is important to be familiar with not only the expertise people have but also how that expertise applies to the task. Our study focuses more specifically on familiarity with team members rather than familiarity with roles. Consequently, we consider a single role in the context of our study which is familiar to all team members in our sample—i.e., software developer. Harrison et al. (2003) have suggested that team member familiarity improves performance by reducing uncertainty about team members and reducing process losses because people spend less time acquiring information about each other. They further argued that the communication flows in familiar teams are more synchronized and more firmly established. They concluded that team members' familiarity with each other provides a basis for coordination

and division of labor and also reduces the need for team interaction. This suggests that team familiarity may be more beneficial for performance in the presence of factors that increase uncertainty about team members, create more process losses, make it difficult to communicate, or make it difficult to coordinate and implement effective team processes.

We posit that team familiarity will have a stronger effect on team performance under more complex team coordination conditions that make it more difficult for team members to communicate, coordinate, and develop further familiarity with each other. Team familiarity should reduce the subjective complexity experienced by team members (Campbell 1988) when they collaborate with each other because they can locate expertise in the team more effectively and figure out more productive ways to work together. Higher team coordination complexity conditions that limit the opportunities for team members to interact and become more familiar with each other or make the available interaction mechanisms less effective will make team familiarity more important for performance. One important team coordination complexity factor often discussed in the software engineering and organizational literatures is geographic dispersion (Carmel 1999, Cohen and Gibson 2003, Herbsleb and Grinter 1999, Herbsleb and Mockus 2003, Kirkman and Mathieu 2005, Powell et al. 2004).

Both the frequency (Allen 1977) and timeliness (Gittell 2001, Waller 1999) of communication can be adversely affected when team members are not in close proximity, often channeling communication into less interactive media with fewer contextual references. Geographic dispersion also affords less shared context for task work (Cramton 2001, Hinds and Mortensen 2005), less shared understanding (Hinds and Weisband 2003) and common ground in team member communication (Olson and Olson 2000), and fewer opportunities for informal and spontaneous communication (Kiesler and Cummings 2002), all of which affect team members' ability to coordinate. In addition, geographic separation causes increased coordination overhead and more substantial delays when initiating contacts and resolving issues (Carmel 1999, Herbsleb and Mockus 2003).

Team familiarity can help geographically dispersed team members mitigate some of these problems by reducing the need to communicate frequently. As Griffith et al. (2003) noted, when distributed teams have formed some collective knowledge, they are better able to access that knowledge. In addition, team familiarity can help team members anticipate each others' actions, develop more effective routines, and more accurately convey and interpret meaning when they do communicate because of their common ground. For example, two geographically dispersed software developers who are familiar with each other and are working on the same modification will be able to anticipate each other's changes more accurately and communicate key integration issues more effectively. As Herbsleb and Grinter (1999) found, geographically distributed teams often use strategies intended to increase familiarity among team members to overcome the problems associated with distance. These strategies include meeting face-to-face before project work begins, assembling teams with members who have worked together in the past, and traveling frequently to other sites. On the other hand, while team familiarity is still beneficial for colocated software teams, it is not as critical because members who are not familiar with each other can always see and talk to each other to coordinate and exchange information about their work (Kraut and Streeter 1995, Perry et al. 1994), making team familiarity less important. As studies of software teams have found, being in close proximity particularly when teammates are both visible and available helps members coordinate their work more effectively through frequent interaction (Teasley et al. 2002). Thus, we posit:

HYPOTHESIS 3. *Team familiarity and geographic dispersion interact positively in their effect on team performance, such that the positive effect of team familiarity on team performance is stronger when teams are geographically dispersed.*

Another important team coordination complexity factor is team size (Brooks 1995, Herbsleb and Mockus 2003, Hinds and Mortensen 2005, Kirkman and Mathieu 2005, Powell et al. 2004, Steiner 1972). Although the effects of team size on performance for traditional colocated teams are well-known, a recent review of the research literature concluded that it is not clear how team size affects team performance for virtual teams (Steiner 1972, cited in Powell et al. 2004). While adding members to a software team contributes more individual productive resources to the team, it also makes it more difficult for all team members to interact and work together to integrate their individual work into a single working product. For example, a team of n members can have as many as $n(n-2)/2$ dependency links. Therefore, the number of possible dyads that need to interact and share information in a team increases exponentially with team size (Brooks 1995).

Highly interdependent tasks like software development generally require more substantial information sharing and team communication (Herbsleb and Grinter 1999) than simpler experimental tasks. Each dependency link between developers will demand time and effort that can be mitigated when members are familiar with each other and can communicate efficiently and effectively. Also, as the team increases in size, the amount of familiarity that an additional team member needs to have or acquire about existing members will be higher. It is more difficult to know which members know what in larger teams, but this type of knowledge is important for performance (Faraj and Sproull 2000), so team familiarity

will be more beneficial for larger teams. For example, a small software team with three developers may not need as much prior familiarity among team members, because each developer can quickly acquire familiarity with the other two. In contrast, a team of 15 developers will have more difficulty acquiring widespread knowledge of who knows what during the task, so prior team familiarity will be more important. Therefore, we posit that:

HYPOTHESIS 4. *Team familiarity and team size interact positively in their effect on team performance, such that the effect of team familiarity on team performance is stronger when teams are larger.*

How Team Familiarity Complements Task Familiarity for Team Performance

So far, we have argued that task and team familiarity interact with task complexity and team coordination complexity in their effect on performance. It is also possible that these two types of familiarity interact with each other such that one may either enhance or substitute for the effect of the other. As Goodman and Leyden (1991) noted, it is not clear whether task and team familiarity have additive or complementary effects on team performance. As discussed earlier, task familiarity helps team members improve performance on individual aspects of their task while team familiarity helps them interact more effectively. While both kinds of familiarity have been associated positively with team performance (Harrison et al. 2003), there may be aspects of each type of familiarity that overlap with the other, such as reducing the difficulty experienced by the members involved or improving common ground for member communication, thus making task and team familiarity somewhat substitutable. On the other hand, experimental studies have shown that when team members develop familiarity with each other, they also develop stronger generalized knowledge about the task domain (Lewis et al. 2005). Similarly, when multiple team members become familiar with the current task and similar prior tasks, they also develop shared schemas that help them form more accurate expectations about what needs to be done and how individual task activities will affect the activities of others, which helps them coordinate implicitly (Cannon-Bowers et al. 1993, Klimoski and Mohammed 1994). While team familiarity primarily affects how team members work together, members with strong team familiarity will also develop some form of collective sense making, enabling them to perform their individual activities more consistently with the team goals and the needs of other team members (Weick and Roberts 1993). These arguments suggest that while task and team familiarity have distinct effects, they may also complement each other.

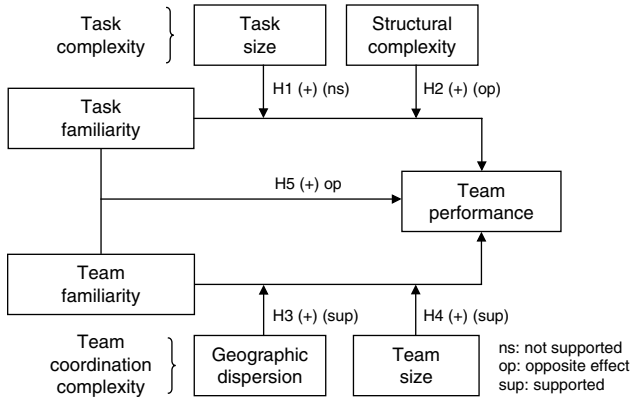
Prior research in information processing provides support for a complementary relationship between team and task familiarity, indicating that when team members share task information, they discuss and use their joint knowledge more effectively because they can recognize each other's expertise (Stasser et al. 1995). For example, Littlepage et al. (1997) found an effect of task familiarity on team performance in all three of their experimental studies with problem-solving teams but, surprisingly, did not find an effect of team familiarity in two out of the three. They speculated that team familiarity did not affect team performance when prior task familiarity was not relevant for the new task. When the task was modified such that familiarity from prior tasks was relevant, they found a significant effect of team familiarity, suggesting that prior task familiarity that is applicable to similar future tasks may enhance the effect of team familiarity.

In software development tasks, while individual expertise with the software helps developers code faster (Brooks 1995), a team of experienced developers who are very familiar with the existing application domain will help members develop views of the task that are more consistent with each other (Curtis et al. 1988). Experienced developers are more likely to be thoroughly familiar with the software processes and tools used by the organization; therefore, they can produce code more efficiently. Similarly, when software developers know each other well and understand each other's needs, their collective sense making enables them to perform their individual tasks in ways that take into account the task needs of other team members (Crowston and Kammerer 1998). This leads us to posit that:

HYPOTHESIS 5. *Task familiarity and team familiarity are complementary in their effects on team performance, such that the positive effect of task familiarity is enhanced when team familiarity is strong and, similarly, the positive effect of team familiarity is enhanced when task familiarity is strong.*

Our study hypotheses are illustrated in our research framework in Figure 1.

Figure 1 Research Hypotheses



Study Method

Research Setting

We collected and analyzed archival data on software development teams from software production sources at a large telecommunications firm. As noted earlier, the software product development setting is particularly well-suited to study familiarity, complexity and team performance. Large software organizations often employ developers with skills ranging from novice levels to experienced developers. Among the experienced developers, some may not be familiar with the company's software code while others are thoroughly familiar with the software system base. Therefore, knowing when familiarity has its strongest impact on performance can help organizations assign personnel to software projects more effectively. In addition, organizations developing software products often operate in highly competitive environments in which time to market is critical, so understanding how familiarity can help reduce software development time can have a substantial financial impact. Nevertheless, performance continues to be problematic in software development due to the many complexities of software tasks such that many projects are not finished on time (Mann 2002).

Archival research is well-suited to performance studies because the data are objective and are unaffected by response bias or response rates. The teams in our study developed a major software product for telephony equipment from two main locations, the United Kingdom and the United States. The similarity in language and culture between the two locations reduces possible confounding effects of these and other factors associated with geographic distance (Espinosa et al. 2003). This software product has been developed incrementally over several years and it contains several million lines of code. Product updates were implemented through numerous "modification request" (MR) projects, each incorporating new and/or updated features into the product base.

An MR project is composed of several individual software changes called "deltas." A delta contains changes of a few lines of code in a single software file made by a single developer. We selected the MR project development team as our unit of analysis because an MR project represents a well-defined unit of software work in the firm we studied, and MR project teams have well-defined membership and very clear goals. Once approved by the organization's "change control board," the MR project is assigned a priority level, a development team, a budget, and other resources. All recorded software production data were traceable to specific MR projects, enabling us to collect useful software production statistics. We studied teams developing software for one large subsystem (over 4 million lines of code) of the telephony product—common channel signaling. This subsystem belonged to a single internal organization, eliminating the potential confounding effects of internal organizational boundaries in teams (Espinosa et al. 2003). The MRs in our sample contained from 2 to 68,000 lines of code and from 1 to 1,050 deltas; they spanned from 1 to 130 modules.

We collected data from software production information recorded for each MR project in the software's configuration management system. Such data sources are often used in studies involving software teams (Herbsleb and Mockus 2003, Kemerer and Slaughter 1999, Mockus and Herbsleb 2002). Our sample consists of all MRs completed for this subsystem during the three years prior to the study in which two or more developers were involved, so only MRs completed by teams are analyzed. The three-year timeframe was selected because no substantial technological changes took place during the period. All MRs produced for the subsystem during this time period were identified and included a total of 1,170 MRs containing 54,665 deltas. The information recorded for each MR project and delta included when the MR project was opened and when a delta was implemented, who worked on it, how many lines of code were developed, and which files and modules were impacted.

Study Variables

Table 1 presents the descriptive statistics and correlation matrix for the study variables. We first define our measures for the main variables of interest: team performance, task familiarity, team familiarity, task complexity, and team coordination complexity. We then define the measures for all control variables. Our variable measures are also summarized in the appendix.

Team Performance. In telecommunications, quality and time to market are critical for sustainable competitive advantage. Therefore, we use the length of time to complete an error-free MR project to measure team performance. We obtained MR project development time by computing the elapsed time between the first and last deltas recorded for a given MR project. A Q-Q plot for this data revealed that this variable was skewed to the left. Several Box-Cox transformations were investigated, and the log transformation gave the best approximation to a normal distribution. A natural logarithm transformation of the performance measure was therefore employed in our analysis. Because performance is improved when development time is reduced, we reversed the sign of this variable (by multiplying it by -1) to make it easier to interpret results, i.e., higher values represent higher levels of performance. Thus, positive regression coefficients represent positive effects on team performance.

Task Familiarity. Prior empirical studies have measured task familiarity simply as prior work experience in a similar task (Littlepage et al. 1997, Reagans et al. 2005). To be effective in their work, software developers need to understand the existing software product base and the application domain in which the new software

Table 1 Descriptive Statistics and Correlation Matrix

Variable	Mean	Std. dev.	1	2	3	4	5	6	7	8	9	10	11
1 Performance	−2.90	1.86											
2 Task familiarity	5.31	1.45	0.19										
3 Team familiarity	1.80	2.95	0.21	0.39									
4 Task size	1.15	3.42	−0.18	0.09	0.03								
5 Structural complexity	3.50	4.44	−0.21	0.00	−0.02	0.14							
6 Team size	2.42	0.88	−0.21	−0.03	−0.10	0.20	0.12						
7 Geographic dispersion	0.10	0.29	−0.08	−0.01	−0.02	−0.03	0.03	0.05					
8 Repairs	0.40	0.49	−0.13	0.00	0.06	0.00	0.02	0.04	0.08				
9 MR project type	0.48	0.50	−0.22	−0.12	−0.15	0.09	0.19	0.19	−0.04	−0.03			
10 System age	1.67	0.88	0.00	0.25	0.18	0.01	−0.04	−0.01	0.05	0.02	−0.12		
11 MR project priority	2.76	0.78	0.17	0.04	0.04	−0.12	−0.16	−0.17	0.03	−0.05	−0.55	0.01	
12 Effort distribution	0.71	0.16	0.28	0.09	0.09	−0.24	−0.19	−0.33	−0.06	−0.06	−0.17	0.00	0.18

Notes. $r > 0.08$ is significant at $p < 0.001$; $r > 0.07$ is significant at $p < 0.01$; $r > 0.05$ is significant at $p < 0.05$.

will operate (Boehm 1981, Brooks 1995, Curtis et al. 1988, Walz et al. 1993). Thus, consistent with similar measures of expertise used in prior software studies (Mockus and Herbsleb 2002), we measured task familiarity by counting the number of deltas developed for the subsystem by each team member prior to starting work on the MR project, and then averaged this count for the entire team. While we could have measured task familiarity at the module or file level, we used the subsystem level because our data showed that these developers generally work on different files and modules rather than building up expertise in a specific set of files or modules. Furthermore, the developers in our study had to be familiar with the subsystem as a whole because the software task activities are highly interdependent; developers must understand how their changes in one file or module will affect other files, modules, and the subsystem as a whole once all parts are integrated and working together.[1]

Because some developers complete more deltas than others for a given MR project, we computed a weighted average for the team using the proportion of deltas completed by each team member in the MR project as the weights. This weighting ensures that the task familiarity experience of developers who completed more deltas in the MR project counts more heavily than that of developers who completed fewer deltas. A Q-Q plot for this data revealed that this variable was skewed to the left. Several Box-Cox transformations were investigated and the log transformation gave the best approximation to a normal distribution, so we employed a natural logarithm transformation of the task familiarity measure.

Team Familiarity. Team familiarity has been measured as prior work experience with the same crew (Goodman and Leyden 1991, Kanki and Foushee 1989), prior knowledge of other team members (Gruenfeld et al. 1996, Harrison et al. 2003), and prior work experience with the same teammates in similar prior tasks (Hinds et al. 2000, Littlepage et al. 1997, Reagans et al. 2005). Developers who have worked on the same software projects will have more shared knowledge about the task domain and about each other than those who have not. Therefore, by counting the number of MRs in which both members had developed code in the past, we first calculated a member familiarity measure for each dyad in each team based on the shared experience that those two developers had with prior MRs. We then aggregated these dyadic measures to the team level by averaging the corresponding shared expertise measures for all dyads that worked on the MR project. The intraclass correlation (ICC) statistic for this variable was 0.745 ($p < 0.001$) indicating that aggregating to the team level is appropriate (Kenny and LaVoie 1985).

Task Complexity

Task Size. We measured task size as the number of thousands of lines of software instructions (i.e., software code added, deleted, or changed) written for the MR. As the software product was written in one programming language, lines of code provides a reasonable measure of software size. Naturally, we expect that more software instructions will take longer to develop.

Structural Complexity. Structural complexity can be evaluated by measuring characteristics that make software difficult to understand and change (Curtis et al. 1979). As the number of modules affected increases, it becomes more difficult to understand how the parts being modified will affect other parts of the system (Darcy et al. 2005, Herbsleb and Mockus 2003) which increases the amount of information that developers need to process to implement the MR, thus increasing its complexity (Wood 1986). Therefore, we measured the structural complexity of the task as the number of modules impacted by the MR. Because each delta affects a single module, information about which modules were affected by an MR is readily available in our data set. We expect that more complex software will take longer to develop because it requires more effort to comprehend and update the code (Banker et al. 1993, Banker et al. 1998).

Team Coordination Complexity

Team Size. We measured team size as the number of developers who completed deltas in the MR project. Because MRs involve very technical and focused modifications, we expect that larger teams will be less efficient at implementing the MR due to the increased need for and difficulty of coordinating, thus taking longer to finish the MR project.

Geographic Dispersion. Consistent with prior studies of global software teams (Herbsleb and Grinter 1999, Herbsleb and Mockus 2003) and because most MRs were developed from either one or two locations (i.e., the United States and the United Kingdom), this variable was dichotomized: 0 if all developers who completed deltas in an MR project worked in the same location and 1 otherwise. Because our study involved only two sites, our measure of dispersion is equivalent to others recommended in the literature, such as counting the number of sites represented in a team (O'Leary and Cummings 2007).[2]

Interaction Effects. We operationalized five interaction variables necessary to test: the interactive effects of task familiarity with the two task complexity variables (task size and structural complexity); the interactive effects of team familiarity with the two team coordination complexity variables (team size and geographic dispersion); and the interactive effects of task familiarity with team familiarity. Interaction variables are constructed by multiplying the respective main variables, which works well when one of the variables in the interaction term is binary or has a few discrete values. However, when both variables are continuous, problems of high multicollinearity can arise. Also, main effect coefficients become difficult to interpret because they represent the value of their main effect when the value of the other variable in the interaction term is zero (Jackard and Turrisi 2003). This creates problems for variables like "team size" that have no meaning when their value is zero.

These difficulties are corrected by centering the dependent variable and the continuous variables used in interaction terms with respect to their means and using these centered variables to construct the interaction terms, which is particularly necessary for variables that have no meaning when they assume a value of zero (Aiken and West 1991). The coefficient of the centered main effect variable represents the magnitude of the main effect when the main variable is at its mean, which has a more meaningful interpretation. We centered the task familiarity variable because a task familiarity value of zero has little meaning in our study; the vast majority of the developers in our sample had developed deltas in the past. Therefore, the coefficient on all main effect variables that contain an interaction term with the task familiarity variable represents their respective effects when task familiarity is at its mean, and the respective interaction terms represent how these effects change as the task familiarity value departs from the mean. We did not center the team familiarity variable because a value of zero is of interest in our study and several teams had no prior team familiarity coming into an MR project. So the coefficient of all main variables that contain an interaction term with team familiarity variable represents their respective effect when team members have not worked together before, and the respective interaction terms represent how these effects change as the team familiarity value departs from zero.

Control Variables. We included five control variables to account for other factors identified in the software engineering literature that can affect software development time: repairs, MR project type, system age, MR project priority, and effort distribution within the MR project team.

Repairs. Defective systems generally take longer to enhance or modify because of the need to discover and repair errors. Deltas in an MR project are recorded as either new code or repair deltas. The percentage of repair deltas in an MR is a good indicator of errors that were found in the software during inspections and tests. The "repair" variable was dichotomized because there were many MRs with no repair deltas (165 MRs or 14.1%), such that the variable was set to a value of 1 if the MR contained more than 38.4% repair deltas (i.e., median value) and 0 otherwise.

MR Project Type. When an MR project is opened, it is classified as either new feature development or feature update. Update MRs modify the functionality of an existing feature while new feature development MRs add new functionality requested by clients or design teams. New feature MRs have uncertainties associated with understanding the requirements of a new product, which will increase software development time. Therefore, we control for project type using a binary variable equal to one for new feature development MRs and zero for feature update MRs.

System Age. We included a variable to control for the effect of time-related factors on software development performance (e.g., technology changes, management changes, etc.). This control variable is also important because the base software product has been growing incrementally for over ten years as new features have been added. We expect that it has become increasingly difficult to integrate the new or updated features into the product base as it has grown. Therefore, we control for time effects using the time elapsed between the start of the first MR project in the product (start date $= 0$) and the current MR's start date, measured in years. Higher numbers represent an older product (i.e., more recent MR projects).

MR Project Priority. When the change control board at the organization approves an MR project for implementation, it also assigns a priority level ranging from 1 to 4. Priority 1 is assigned to MRs that affect critical client services, whereas priority 4 is assigned to MRs involving nuisance problems that need to be corrected but can wait. Naturally, we need to control for MR project priority because more critical MRs receive more attention than lower priority MRs. We have reversed this scale for ease of interpretation of results, such that 1 represents the lowest priority and 4 the highest; i.e., higher values of this variable are associated with higher priority levels.

Effort Distribution. We measured the distribution of effort in an MR project team using a measure of evenness in the contribution of deltas to the project. An MR project in which one developer does most of the coding may take longer than one in which every developer contributes equally because the developers can work in parallel and finish sooner. This variable was constructed using a Gini coefficient of homogeneity (Alker and Russet 1964, Dorfman 1979, Watson and Finholt 1986), ranging between zero (one person develops all deltas) and one (each developer contributes an equal number of deltas).

Data Analysis and Results

We analyzed the data using ordinary least-squares regression methods. Our results are summarized in Table 2. We inspected our model for multicollinearity by computing a condition index (Belsley et al. 1980) for the entire model and variance inflation factors (VIF) (Marquardt 1970) for each of the independent variables (see collinearity statistics in Table 2). We also conducted Durbin-Watson's test for autocorrelation and White's test for heteroskedasticity (Greene 1997, Kennedy 1992). None of these tests revealed any diagnostic problems.

Hierarchical Regression

Consistent with standard practice for analyzing models with interaction effects (Aiken and West 1991, Cohen and Cohen 1983), variables were entered in the regression model in blocks in a hierarchical fashion. A hierarchical regression model helps us evaluate whether the familiarity and interaction variables add significant explanatory power to the model incrementally over all other variables. Therefore, we included only task complexity, team coordination complexity, and control variables in the "*baseline model*." We then entered the familiarity variables into the "*familiarity model*," which significantly increased the explanatory power of the regression model ($\Delta R^2 = 0.047$, $F_{R^2\ \text{Change}} = 32.256$, $p < 0.001$), suggesting that team and task familiarity help explain significant variance in team performance incrementally over what task complexity, team coordination complexity, and control variables explain. Finally, we entered all the interaction variables into the "*interaction model*," which further increased the predictive power of the regression model ($\Delta R^2 = 0.031$, $F_{R^2\ \text{Change}} = 8.979$,

Table 2 Regression Model on Team Performance

								Hypotheses		
	Baseline model		+ Familiarity vars		+ Interaction vars				Effects	
Variable	Coefficient	*P*-value	Coefficient	*P*-value	Coefficient	*P*-value	Coll VIF	No.	Pred	Result
Task size	−0.044	0.003	−0.057	<0.001	−0.051	0.008	2.119			
Structural complexity	−0.043	<0.001	−0.046	<0.001	−0.066	<0.001	1.348			
Team size	−0.171	0.005	−0.157	0.008	−0.282	<0.001	1.714			
Geographic dispersion	−0.383	0.018	−0.344	0.029	−0.422	0.013	1.228			
Repairs	−0.322	0.001	−0.348	<0.001	−0.364	<0.001	1.038			
MR project type	−0.489	<0.001	−0.375	0.001	−0.303	0.009	1.532			
System age	−0.074	0.184	−0.182	0.001	−0.167	0.003	1.125			
MR project priority	0.073	0.340	0.095	0.201	0.126	0.086	1.461			
Effort distribution	2.000	<0.001	1.731	<0.001	1.594	<0.001	1.243			
Task familiarity			0.163	<0.001	0.364	<0.001	3.351			
Team familiarity			0.082	<0.001	0.182	<0.001	3.582			
Task familiarity × task size					0.004	0.744	2.004	1	+	n.s.
Task familiarity × structural complexity					−0.047	<0.001	3.259	2	+	Opposite
Team familiarity × geogr dispersion					0.104	0.012	1.503	3	+	Supported
Team familiarity × team size					0.106	0.002	2.101	4	+	Supported
Team familiarity × task familiarity					−0.069	<0.001	3.802	5	+	Opposite
N		1,110		1,110		1,110				
Adjusted R^2		0.151		0.197		0.225				
R^2		0.158		0.205		0.236				
Change in R^2		0.158		0.047		0.031				
F test for change in R^2		23.01		32.256		8.797				
p-value of F test		<0.001		<0.001		<0.001				
Condition index (collinearity)		19.773		19.964		20.610				

$p < 0.001$), suggesting that the interaction variables explain significant variation in team performance over that explained by the familiarity and other variables.

Results of Baseline Model

The results of the baseline model show that the effects of all control variables were generally as expected, providing some assurance of the validity of our model. MRs that contained a higher proportion of repair deltas were associated with reduced performance ($\beta = -0.322$, $p < 0.001$), i.e., took longer to develop, most likely because it takes more time to identify, analyze, and correct errors in the system. Similarly, new feature development MRs had lower performance ($\beta = -0.489$, $p < 0.001$). As noted earlier, this effect is likely due to the added difficulty of learning, understanding, and incorporating new system and client requirements related to these new features. Also, an even distribution of development effort within the MR project team was associated with higher performance ($\beta = 2.000$, $p < 0.001$) because software is developed faster when the developers work in parallel than when one or two developers do the majority of the work. We found no effect of system age and MR project priority on software development time in the baseline model but, as we discuss later, these effects became significant once the familiarity and interaction variables were entered into the model. Our results show that task complexity also had a main effect on performance. As expected, we found that MR projects that develop larger ($\beta = -0.044$, $p = 0.003$) and more structurally complex software ($\beta = -0.043$, $p < 0.001$) were associated with reduced performance. Similarly, both team coordination complexity variables had a main effect on performance. Team size ($\beta = -0.171$, $p = 0.005$) and geographic dispersion ($\beta = -0.383$, $p = 0.018$) were associated with reduced performance.

Results of the Familiarity Model

Except for system age, the direction and significance of the coefficients on the other variables in the baseline model did not change substantially when the familiarity variables were added. The effect of system age on performance in the baseline model was negative but nonsignificant; it became significant in the familiarity model ($\beta = -0.182$, $p = 0.001$), indicating that for given levels of task and team familiarity, more recent MRs are associated with lower performance. The results of the familiarity model also show that task familiarity had a significant effect on software development time. Both task familiarity ($\beta = 0.163$, $p < 0.001$) and team familiarity ($\beta = 0.082$, $p < 0.001$) had a positive effect on performance. This result provides some assurance of the validity of our measures and results because they are consistent with prior findings in familiarity research.

Results of Interaction Model

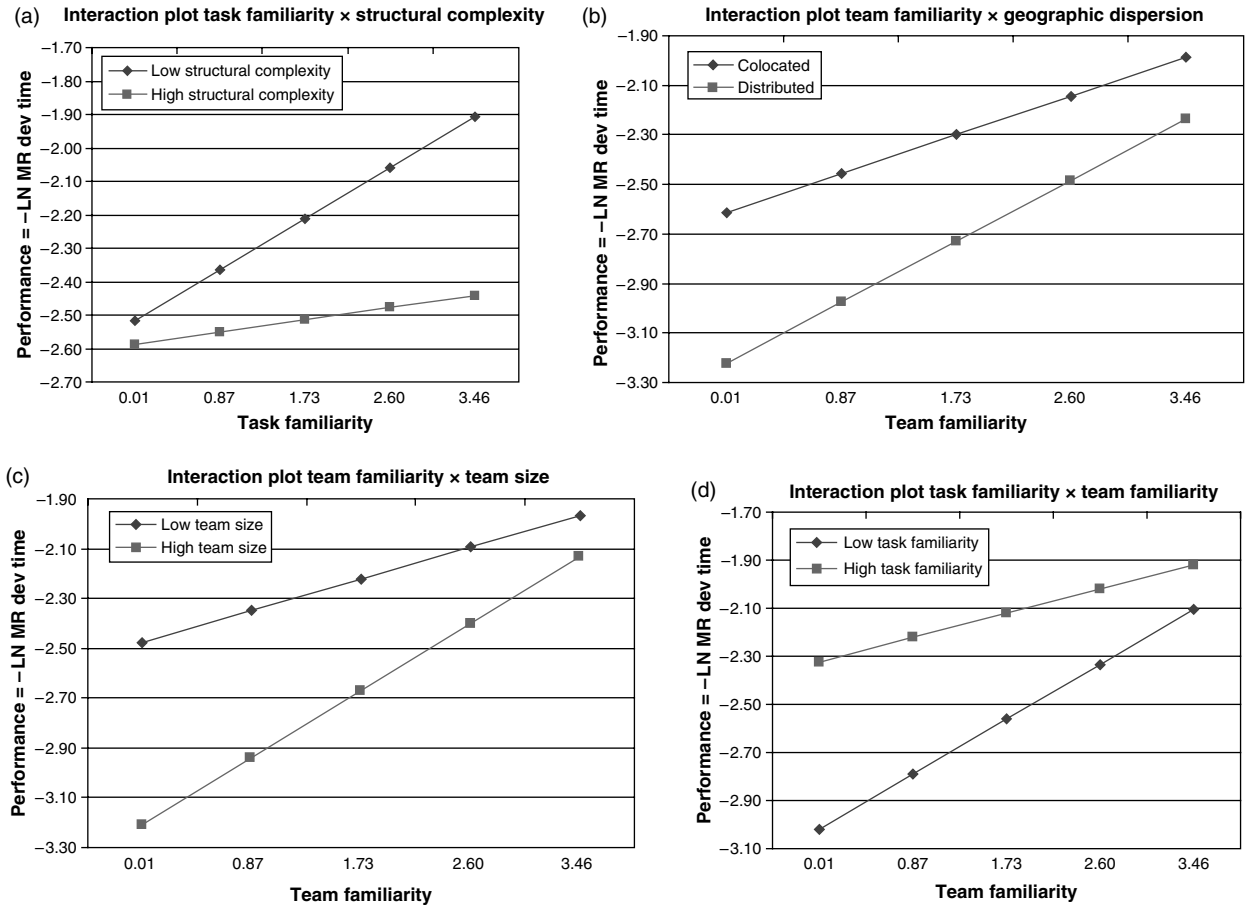
The direction and significance of all coefficients did not change substantially when the interaction variables were added to the familiarity model. Only the effect of MR project priority changed slightly, becoming marginally significant ($\beta = 0.126$, $p = 0.086$) and in the expected direction, indicating that higher priority MRs are associated with higher levels of performance. The interaction effects are illustrated in Figure 2.[3] With respect to the interactive effects of task familiarity and task complexity, we found, contrary to our expectations, that the effect of task familiarity on performance was unaffected by task size. Therefore, Hypothesis 1 was not supported. For Hypothesis 2, while we found that the effect of task familiarity on performance was affected by structural complexity ($\beta = -0.047$, $p < 0.001$), the sign of the coefficient on the interaction effect was in a direction opposite from our expectation. As Figure 2(a) illustrates, the positive effect of task familiarity on performance was stronger for tasks with lower structural complexity.

On the other hand, we did find that the effect of team familiarity on team performance was enhanced when team coordination complexity was higher, supporting Hypotheses 3 and 4. Team familiarity improved performance more strongly with larger teams ($\beta = 0.106$, $p = 0.002$) and with geographically dispersed teams ($\beta = 0.104$, $p = 0.012$), who took an average of 97.3 days to implement MRs compared with 48.2 days for colocated teams. As illustrated in Figures 2(b) and 2(c), the difference in performance between colocated and geographically dispersed and between small and large teams narrowed substantially as team familiarity increased. Finally, as we hypothesized, we found that team familiarity and task familiarity had a significant interaction effect on team performance ($\beta = -0.069$, $p < 0.001$). However, this effect was in the opposite direction from our expectation, so Hypothesis 5 was not supported. As Figure 2(d) illustrates, the performance differential for teams with high task familiarity relative to teams with low task familiarity diminished substantially as team familiarity increased and vice versa, suggesting that the effects of task and team familiarity are more substitutive than complementary.

Discussion

Our study represents an important and original contribution to the research literature on organizational team performance and familiarity by providing insight into how familiarity complements or offsets aspects of task complexity for distributed teamwork. Consistent with prior research on software engineering (Boehm 1981, Brooks 1995, Curtis et al. 1988, Walz et al. 1993) and familiarity (Harrison et al. 2003, Littlepage et al. 1997), our findings confirmed that both task and team familiarity had positive effects on performance. It is noteworthy that our

Figure 2 Interaction Diagrams



findings are the first to reinforce this empirical evidence with organizational teams and objective data sources. Perhaps the most significant contribution of our study is a better understanding about how different dimensions of familiarity and complexity interact with each other such that when a given dimension is excessive or absent, team members can resort to other dimensions to attain high levels of performance. Our findings provide evidence that when teams face complexity in a particular dimension of their work, they can resort to team familiarity to reduce the subjective complexity they will experience when working with each other. For example, team members can increase their familiarity with each other to reduce their coordination complexity. In addition, when team members lack team familiarity they can resort to task familiarity to get the work done and vice versa. However, our results show that the inherent complexity of certain tasks is irreducible, such that the beneficial effects of increased task familiarity experience diminishing returns when task complexity increases. We elaborate on these findings below.

Task familiarity had mixed interactive effects with task complexity. Contrary to our expectations, the effect of task familiarity on team performance was unaffected by task size. One possible explanation may be that there are inherent complexities in software tasks associated with the size of the software product and these complexities are irreducible. Therefore, as the size of the task increases, software development time increases proportionally and, as task familiarity increases, software development time decreases proportionally, but there are no dramatic productivity improvements possible to compensate for increased software product size that can be attributed to task familiarity. Furthermore, because large-scale software development organizations often use configuration management tools to archive the software components created and to document technical issues about the software modifications (Grinter 2000), larger software tasks may not necessarily benefit from additional task familiarity because much of the technical information that developers may need is readily available in these tools.

Also contrary to our expectations, and as Figure 2(a) illustrates, we found that task familiarity helped reduce task completion time more strongly for tasks with lower structural complexity. While developing more complex software that spans more modules still takes longer than developing simpler software spanning fewer modules, dramatic productivity improvements do not appear possible for more structurally complex tasks through task

familiarity alone; perhaps more specialized expertise in particular domains, functions, or technologies may also be necessary. Our results suggest that a unit of familiarity increase has a stronger impact on structurally simpler tasks, perhaps because the experience gained from repetition can have dramatic performance benefits. However, as the structural complexity of the task increases to irreducible levels, higher task familiarity alone can not yield dramatic improvements in performance.

To further explore this issue, we split the sample at the median value of task familiarity into high and low task familiarity groups and conducted an analysis of variance for various variables. We found that high task familiarity teams worked on larger software ($F_{1,1287} = 14.420$, $p < 0.001$) affecting more files ($F_{1,1287} = 7.659$, $p = 0.006$). They also tended to work more on feature updates than new features ($F_{1,1287} = 6.836$, $p = 0.009$). High and low task familiarity groups worked on MRs spanning a roughly similar number of modules, i.e., similar structural complexity. So, when structural complexity increases in the high task familiarity groups who often work on larger software and feature updates, the positive effect of their task familiarity may diminish because the problems they face may require more innovation and critical thinking than business as usual. At the same time, when structural complexity increases in the low familiarity groups who often work on smaller software that adds new functionality, the positive effect of their task familiarity may experience diminishing returns because it is less relevant for the understanding of new requirements.

These possible explanations suggest that the effect of task familiarity on performance may be moderated more by the source or type of complexity than by the level of complexity. For example, the complexities inherent in the existing product base may be adequately handled by developers who are familiar with the software, but if the complexity comes from sources external to the existing code (e.g., new regulations, new client requirements, new technologies), then task familiarity with the current subsystem may actually hinder the developers' ability to innovate. This explanation would be consistent with prior findings in experimental studies of task experience and organizational learning, suggesting that simpler tasks that require less innovation are more positively affected by task experience because this knowledge is more applicable to similar simple tasks (Argote et al. 1995). We speculate that the developers' expertise in areas other than the current subsystem (e.g., functional expertise with other software or with other technologies) may help more with complex software than does general task domain familiarity.

With respect to team coordination complexity, we found that geographic dispersion and team size had a negative effect on performance, which is consistent with prior research on distributed software teams (Carmel 1999, Herbsleb and Mockus 2003), virtual teams (Armstrong and Cole 2002), and software development (Brooks 1995). Consistent with our expectation, we also found that team familiarity helped to mitigate these negative effects. As illustrated in Figure 2(b), team familiarity helped narrow the performance difference between colocated and geographically dispersed teams. When teammates are familiar with each other, regardless of location, they know whom to contact for answers to their questions and they may obtain cooperation and responses to their queries more quickly. Without familiarity, geographically dispersed team members need to bridge distance and technology-mediated boundaries (Hinds and Bailey 2003) to coordinate their work in some other way, making it more difficult to work together because they do not enjoy the benefits of copresence—e.g., presence awareness, frequent communication, and contextual reference.

As illustrated in Figure 2(c), team familiarity also helped narrow the difference in performance between small and large teams. This finding is important because the literature suggests mixed effects of team size on performance. On one hand, more members help performance because they represent additional productive resources in the team. On the other hand, larger teams cause increased coordination overhead due to the exponentially larger number of possible interaction links between members, which can hurt performance, i.e., Brooks' "man month" myth (1995). Our results show conclusively that, all else being equal, team size decreases performance when team members are not familiar with each other. However, familiarity mitigates the negative effect of team size on performance. By the same token, our results show that team familiarity is not as important with small teams but becomes critical as team size increases. In sum, the stronger effect of team familiarity on performance for geographically dispersed and larger organizational teams supports our expectation that team familiarity is more beneficial with higher team coordination complexity in which it is more difficult to interact, coordinate, and share information.

Contrary to our expectations, we found that team and task familiarity are substitutive, not complementary in their effects on team performance. As Figure 2(d) shows, task and team familiarity are both beneficial for performance but when one type of familiarity is lacking, the effect of the other type of familiarity is much stronger. Conversely, as either type of familiarity increases, the effect of the other type of familiarity becomes less powerful, suggesting that these two types of familiarity are somewhat substitutable. This finding represents an important contribution to the literature on transactive memory. Recent research has theorized that in order for transactive memory to be effective, team members need to have knowledge about three things: the task (what needs to be done), what type of expertise is required to

perform the task, and the people in the team who possess those skills (Brandon and Hollingshead 2004). However, this body of research is not clear about whether knowledge of the task and the team complement or substitute for each other, or if their respective effects are simply additive. Our findings suggest that when team members have more task familiarity and more team familiarity, performance increases but at a diminishing rate. These results provide empirical evidence that task expertise and directory structure (i.e., knowledge of where to locate this expertise within the team) are both important for performance, but that they appear to substitute for each other as either type of knowledge increases. In other words, having more task expertise makes one less dependent on colleagues, whereas having a broad knowledge of who knows what makes one less dependent on task expertise.

One possible explanation for this result is that deeper levels of familiarity with a shared task may provide team members with knowledge schemas that help them anticipate more accurately how the task will progress and what other team members will be doing. This knowledge will help them to coordinate implicitly and make team familiarity less important. Similarly, deeper levels of team familiarity help team members understand who knows what in the team, thus helping them specialize and figure out where task expertise is located when needed, and thus making them less dependent on their own task knowledge. In contrast, team members who are not very familiar with the task will benefit more from team familiarity because it gives them common ground and common context, helping them interact more effectively and locate and access expertise when needed.

Our results have strong implications for organizational teamwork because they demonstrate that dramatic improvements in productivity are not possible through task familiarity alone because high levels of task complexity appear to be irreducible, so further investments in task experience can only go so far. Furthermore, as team familiarity increases, task familiarity becomes less important because of their substitutive effects. At the same time, team familiarity can have dramatic effects in productivity by reducing the subjective team coordination complexity experienced by its members when working together on a task, particularly for larger and more geographically dispersed teams.

Finally, our study also underscores one general but important methodological issue in familiarity research: Task familiarity can be measured at different levels, but the task familiarity that really matters is that which can be transferred to subsequent tasks. In our study, familiarity with the same files or modules being modified was not as important as familiarity with the entire subsystem because developers worked more often in different files and modules. The interdependent nature of this software task required developers to understand how their changes in one file or module affect other files and modules in the subsystem. Thus, the familiarity relevant for the task was with the subsystem.

Conclusions

Our study makes evident the importance of investigating interaction effects in organizational teams and familiarity research. In this study, we have hypothesized and empirically evaluated how task and team familiarity interact with task and team coordination complexity in their effect on team performance. Our results show that while familiarity is generally beneficial for performance, it can also complement or offset different elements of complex work environments. Analyzing each of these dimensions separately, we first found that both types of familiarity have positive effects on performance. We also found that all dimensions of complexity had negative effects on performance. Both of these findings are consistent with intuition and prior research.

When theorizing about the interaction effects between familiarity and complexity, prior research has argued that familiarity helps performance more when tasks are more complex, but the empirical evidence suggests the opposite because less familiar team members are able to innovate and attain higher levels of performance with complex tasks (Argote et al. 1995). Our study delves deeper into different dimensions of familiarity and complexity and helps resolve these contradictory findings. We studied two important types of familiarity—task and team familiarity; we decomposed task complexity into two components—task size and structural complexity; and then we incorporated a new dimension called "team coordination complexity," representing the additional complexity that needs to be taken into account when the task is carried out collaboratively. We decomposed this into two components—team size and geographic dispersion. When we analyzed the interaction effects of task familiarity with task complexity, we found that our results were similar to prior research for structural complexity—task familiarity is more beneficial for less structurally complex tasks. However, we found no interaction effect with task size. These results suggest that when experienced professionals work on large organizational tasks, these tasks take longer to complete but they are somewhat indivisible in the sense that further task experience does not help offset the negative effects of task size. On the other hand, consistent with intuition, team familiarity does help teams offset the negative effect of team size and dispersion, suggesting that investments made to develop team familiarity (e.g., frequent trips to each other's sites, forming teams with members that have worked together, implementing technologies like video conferencing that foster team familiarity) do pay off.

Finally, the substitutive effect of team and task familiarity on performance represents an intriguing finding,

which tells us that for transactive memory to be effective, teams need to develop both team and task familiarity. At the same time, our findings suggest that the benefits of both types of familiarity are substitutable and that their joint effects experience diminishing returns as both types of familiarity increase.

Our study has some limitations. First, it was conducted in a single organization. Although this focused research design mitigates or eliminates potential confounds and increases the internal validity of our results, they may be bound to the idiosyncrasies of this organization. The findings could be limited to software organizations working on very large products that are updated incrementally and may not extend to other software products and other development methods. Further studies are needed to validate our findings in other contexts. Also, our familiarity measures are obtained from software production records; while the objective data are not subject to the biases and limitations of subjective measures, the data do not measure psychological attitudes and aspects of team processes.

While our findings are limited to software teams, we believe that they could generalize to other collaborative, intellectual, and technical tasks involving knowledge workers and containing highly interdependent activities with varying degrees of complexity. Generally speaking, our findings about the interactive effects of task familiarity should be generally applicable to other types of knowledge work like design engineering, requirements analysis, and technical product development in which increased task complexity can be tackled by innovating rather than by doing more of the same, whereas simple tasks can benefit from more experience with similar tasks. Our findings may not necessarily generalize to tasks in which the application of knowledge acquired in prior tasks and task experience acquired from repetition may be more critical than innovation, such as emergency work and other fast-paced real-time tasks in which there is not much time to innovate. On the other hand, we believe that our findings about the interactive effects of team familiarity are more generally applicable to a wider range of tasks because prior experience with other team members will help members coordinate their work with larger teams and across geographic boundaries.

Finally, the team familiarity dimension in our study only considers familiarity with team members and not familiarity with roles. Our sample only included one role—software developer—and developers in our study context are thoroughly familiar with this role. Further studies need to be conducted to better understand the effects of familiarity with expert roles and how these effects may interact with task and team familiarity and complexity.

Acknowledgments

The authors would like to thank Audris Mockus, F. Javier Lerch, Linda Argote, Ann Majchrzak, and Mildred F. Myers for their most valuable advice and guidance at various stages of this research project. They would also like to thank the reviewers for their excellent feedback and suggestions during the review process.

Appendix. Summary of Variable Measures, Hypotheses, and Results

Variable	Measure
Performance	Length of time to complete an error-free MR project, reversed to measure performance, and log transformed for normality
Task familiarity	Average number of deltas completed by team members, not including the current MR, weighted by the proportion of deltas contributed by each member to the current MR, log transformed for normality
Team familiarity	Average number of MRs in which each dyad worked together, not including the current MR, averaged for all dyads in the team
Task size	Thousands of lines of software code added, changed, or deleted in the MR
Structural complexity	Number of modules affected by the MR
Team size	Number of developers who contributed deltas to the MR
Geographic dispersion	1 if the team included members from two locations and 0 if all members were colocated
Repairs	1 if the MR contains more repair deltas than the median and 0 otherwise
MR project type	1 if the MR was opened to implement new features and 0 if opened to update existing features
System age	Elapsed time between the start of the first MR project in the system (start date = 0) and the current MR's start date, measured in years
Priority	Implementation priority level assigned by the change control board, from 1 (i.e., a nusisance problem) to 4 (top priority affecting critical client services)
Effort distribution	Gini coefficient of homogeneity, from 0 (one developer develops all deltas) to 1 (all developers contribute an equal number of deltas)

Endnotes

[1]We investigated two additional measures of task familiarity at a more microlevel, one for file familiarity and one for module familiarity based on the number of deltas written in the respective file and module. On average, a developer had a task familiarity of 55.3 deltas with the entire subsystem. Also, on average, a developer had modified 5.8 modules in this subsystem and had written 19.0 deltas in the same module as the one in the delta being modified. Finally, on average, a developer had modified 28.1 files in the subsystem and had written 3.0 deltas in the same file as the one being modified in that delta. These numbers suggest that developers often have little familiarity with the particular files they are modifying. At the

same time, developers have some familiarity with the modules they are updating but they have even more familiarity with other modules. While a familiarity of 19.0 deltas with the module being modified is an important component of task familiarity, the developer's familiarity with the remaining 4.8 modules (i.e., 5.8 minus 1) represents important application domain familiarity that needs to be taken into account, which is consistent with the findings of Reagans et al. (2005) that familiarity with other tasks that are related to the current task matters. We further analyzed these alternative measures and found that although these variables are correlated with our current task familiarity measure ($r = 0.234$, $p < 0.001$ with same files and $r = 0.414$, $p < 0.001$ with same modules), neither of these variables had significant main effects on team performance, suggesting that task familiarity at the subsystem level has more predictive power.

[2]We contemplated using a continuous measure of dispersion based on the proportion of distributed dyads in the team, but decided against it because more than half of the values would have been truncated at zero (i.e., when all members were colocated), thus presenting other analytical challenges. Furthermore, this continuous variable was highly correlated with our binary variable ($r = 0.709$, $p < 0.001$) so the results would be quite similar, but binary variables facilitate the interpretation of interaction effects (Aiken and West 1991).

[3]Interaction diagrams are appropriate to illustrate interaction effects of a continuous variable with one binary variable because the slope of the continuous variable in the regression diagram can be shown as two separate values, one when the binary variable takes a value of zero and another when the binary variable takes value of one. However, when both variables in the interaction term are continuous, these diagrams are less useful because the slope of one of the variables changes for every (continuous) value of the other variable (Aiken and West 1991). This is further complicated when the regression model is multivariate. However, the interaction effect can be illustrated by splitting the data at the median of each continuous variable and creating binary variables for the respective high and low values, and by keeping all other variables not included in the interaction term constant at their means. The diagrams in Figure 2 were constructed using this method for illustration purposes only. The upper and lower values of the respective variables represent the means of the continuous variable in the upper and lower side of the median split, respectively.

References

Aiken, L., S. West. 1991. *Multiple Regression: Testing and Interpreting Interactions*. Sage, Newbury Park, CA.

Alavi, M., D. E. Leidner. 2001. Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Quart.* **25**(1) 107–136.

Alker, H., M. Russet. 1964. On measuring inequality. *Behavioral Sci.* **9**(3) 207–218.

Allen, T. 1977. *Managing the Flow of Technology*. MIT Press, Cambridge, MA.

Argote, L., C. A. Insko, N. Yovetich, A. A. Romero. 1995. Group learning curves: The effects of turnover and task complexity on group performance. *J. Appl. Soc. Psych.* **25**(6) 512–529.

Armstrong, D. J., P. Cole. 2002. Managing distances and differences in geographically distributed work groups. P. Hinds, S. Kiesler, eds. *Distributed Work*. MIT Press, Cambridge, MA, 187–215.

Banker, R. D., S. A. Slaughter. 2000. The moderating effects of structure on volatility and complexity in software enhancement. *Inform. Systems Res.* **11**(3) 219–240.

Banker, R., S. Dattar, C. Kemerer. 1993. Software complexity and software maintenance costs. *Comm. ACM* **36**(11) 81–94.

Banker, R., G. Davis, S. A. Slaughter. 1998. Software development practices, software complexity, and software maintenance performance: A field study. *Management Sci.* **44**(4) 433–450.

Belsley, D., E. Kuh, R. Welsch. 1980. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. John Wiley & Sons, New York.

Boehm, B. R. 1981. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ.

Brandon, D. P., A. Hollingshead. 2004. Transactive memory systems in organizations: Matching tasks, expertise, and people. *Organ. Sci.* **15**(6) 633–644.

Brooks, F. 1995. *The Mythical Man-Month: Essays on Software Engineering*, anniversary ed. Addison-Wesley, Reading, MA.

Campbell, D. J. 1988. Task complexity: A review and analysis. *Acad. Management Rev.* **13**(1) 40–52.

Cannon-Bowers, J. A., E. Salas, S. Converse. 1993. Shared mental models in expert team decision-making. J. Castellan, ed. *Individual and Group Decision-Making: Current Issues*. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 221–246.

Carmel, E. 1999. *Global Software Teams*. Prentice-Hall, Upper Saddle River, NJ.

Cohen, J., P. Cohen. 1983. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

Cohen, S. G., D. E. Bailey. 1997. What makes teams work: Group effectiveness research from the shop floor to the executive suite. *J. Management* **23** 239–290.

Cohen, S. G., C. B. Gibson. 2003. In the beginning: Introduction and framework. S. G. Cohen, C. B. Gibson, eds. *Virtual Teams that Work: Creating Conditions for Virtual Team Effectiveness*. Jossey-Bass, San Francisco, CA, 1–13.

Cramton, C. D. 2001. The mutual knowledge problem and its consequences for dispersed collaboration. *Organ. Sci.* **12**(3) 346–371.

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

Curtis, B., H. Krasner, N. Iscoe. 1988. A field study of the software design process for large systems. *Comm. ACM* **31**(11) 1268–1286.

Curtis, B., S. B. Sheppard, P. Milliman, M. A. Borst, T. Love. 1979. Measuring the psychological complexity of software maintenance tasks with the Halstead and Mccabe metrics. *IEEE Trans. Software Engrg.* **5**(2) 96–104.

Darcy, D. P., C. F. Kemerer, S. A. Slaughter, J. E. Tomayko. 2005. The structural complexity of software: An experimental test. *IEEE Trans. Software Engrg.* **31**(11).

Dorfman, R. 1979. A formula for the Gini coefficient. *Rev. Econom. Statist.* **61** 146–156.

Espinosa, J. A., J. N. Cummings, J. M. Wilson, B. M. Pearce. 2003. Team boundary issues across multiple global firms. *J. Management Inform. Systems* **19**(4) 157–190.

Faraj, S., L. Sproull. 2000. Coordinating expertise in software development teams. *Management Sci.* **46**(12) 1554–1568.

Gittell, J. H. 2001. Supervisory span, relational coordination, and flight departure performance: A reassessment of post-bureaucracy theory. *Acad. Management J.* **12**(4) 468–483.

Goodman, P. S., S. Garber. 1988. Absenteeism and accidents in a dangerous environment: Empirical analysis of underground coal mines. *J. Appl. Psych.* **73**(1) 81–86.

Goodman, P. S., D. P. Leyden. 1991. Familiarity and group productivity. *J. Appl. Psych.* **76**(4) 578–586.

Goodman, P. S., S. Shah. 1992. Familiarity and work group outcomes. S. Worchel, W. Wood, J. A. Simpson, eds. *Group Processes and Productivity*. Sage Publications, Newbury Park, CA, 578–586.

Greene, W. 1997. *Econometric Analysis*. Prentice-Hall, Upper Saddle River, NJ.

Griffith, T. L., J. E. Sawyer, M. A. Neale. 2003. Virtualness and knowledge in teams: Managing the love triangle of organizations, individuals, and information technology. *MIS Quart.* **27**(2) 265–287.

Grinter, R. E. 2000. Workflow systems: Occasions for success and failure. *Comput. Supported Cooperative Work* **9** 189–214.

Gruenfeld, D. H., E. A. Mannix, K. Y. Williams, M. A. Neale. 1996. Group composition and decision making: How member familiarity and information distribution affect process and performance. *Organ. Behav. Human Decision Processes* **67**(1) 1–15.

Hackman, R. 1987. The design of work teams. J. Lorsch, ed. *Handbook of Organizational Behavior*. Prentice-Hall, Englewood Cliffs, NJ.

Harrison, D. A., S. Mohammed, J. E. McGrath, A. T. Florey, S. W. Vanderstoep. 2003. Time matters in team performance: Effects of member familiarity, entrainment, and task discontinuity on speed and quality. *Personnel Psych.* **56**(3) 633–669.

Herbsleb, J. D., R. E. Grinter. 1999. Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software* **16**(5) 63–70.

Herbsleb, J. D., A. Mockus. 2003. An empirical study of speed and communication in globally distributed software development. *IEEE Trans. Software Engrg.* **29**(6) 481–494.

Hinds, P., M. Mortensen. 2005. Understanding conflict in geographically distributed teams: The moderating effects of shared identity, shared context, and spontaneous communication. *Organ. Sci.* **16**(3) 290–307.

Hinds, P., S. Weisband. 2003. Knowledge sharing and shared understanding in virtual teams. S. G. Cohen, C. B. Gibson, eds. *Virtual Teams that Work: Creating Conditions for Virtual Team Effectiveness*. Jossey-Bass, San Francisco, CA, 21–36.

Hinds, P. J., D. E. Bailey. 2003. Out of sight, out of synch: Understanding conflict in distributed teams. *Organ. Sci.* **14**(6) 615–632.

Hinds, P. J., K. M. Carley, D. Krackhardt, D. Wholey. 2000. Choosing work group members: Balancing similarity, competence, and familiarity. *Organ. Behav. Human Decision Processes* **81**(2) 226–251.

Jackard, J., R. Turrisi. 2003. *Interaction Effects in Multiple Regression*. Sage Publications, London, UK.

Kanki, B. G., H. C. Foushee. 1989. Communication as group process mediator of aircrew performance. *Aviation, Space, Environ. Medicine* **60**(5) 402–410.

Katz, R. 1982. The effects of group longevity on project communication and performance. *Admin. Sci. Quart.* 27 81–104.

Kemerer, C. 1995. Software complexity and software maintenance: A survey of empirical research. *Ann. Software Engrg.* (1) 1–22.

Kemerer, C., S. A. Slaughter. 1999. An empirical approach to studying software evolution. *IEEE Trans. Software Engrg.* **25**(4) 1–17.

Kennedy, P. 1992. *A Guide to Econometrics*. MIT Press, Cambridge, MA.

Kenny, D. A., L. LaVoie. 1985. Interpersonal relations and group processes. *J. Personality Soc. Psych.* **48**(2) 339–348.

Kiesler, S., J. N. Cummings. 2002. What do we know about proximity in work groups? A legacy of research on physical distance. P. Hinds, S. Kiesler, eds. *Distributed Work*. MIT Press, Cambridge, MA, 57–80.

Kirkman, B. L., J. Mathieu. 2005. The dimensions and antecedents of team virtuality. *J. Management* **31**(5) 1–19.

Klimoski, R. J., S. Mohammed. 1994. Team mental model: Construct or metaphor. *J. Management* **20**(2) 403–437.

Kraut, R. E., L. A. Streeter. 1995. Coordination in software development. *Comm. ACM* **38**(3) 69–81.

Lewis, K. 2003. Measuring transactive memory systems in the field: Scale development and validation. *J. Appl. Psych.* **88**(4) 587–604.

Lewis, K., D. Lange, L. Gillis. 2005. Transactive memory systems, learning and learning transfer. *Organ. Sci.* **16**(6) 581–598.

Liang, D., R. Moreland, L. Argote. 1995. Group versus individual training and group performance: The mediating role of transactive memory. *Personality Soc. Psych. Bull.* **21** 384–393.

Littlepage, G., W. Robison, K. Reddington. 1997. Effects of task experience and group experience on group performance, member ability, and recognition of expertise. *Organ. Behav. Human Decision Processes* **69**(2) 133–147.

Malone, T., K. Crowston. 1994. The interdisciplinary study of coordination. *ACM Comput. Surveys* **26**(1) 87–119.

Mann, C. C. 2002. Why software is so bad. *Tech. Rev.* **105**(5) 33–38.

Marquardt, D. W. 1970. Generalized inverses, ridge regression, biased linear estimation, and non-linear estimation. *Technometrics* **12** 591–612.

McGrath, J. 1991. Time, interaction and performance (tip). *Small Group Res.* **22**(2) 147–174.

Mockus, A., J. D. Herbsleb. 2002. Expertise browser: A quantitative approach to identifying expertise. *24th Internat. Conf. Software Engrg.*, Orlando, FL.

O'Leary, M. B., J. N. Cummings. 2007. The spatial, temporal, and configurational characteristics of geographic dispersion in teams. *MIS Quart.* Forthcoming.

Olson, G. M., J. S. Olson. 2000. Distance matters. *Human-Comput. Interaction* **15**(1) 139–179.

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

Powell, A., G. Piccoli, B. Ives. 2004. Virtual teams: A review of current literature and directions for future research. *Data Base Adv. Inform. Systems* **35**(1) 6–36.

Reagans, R., L. Argote, D. Brooks. 2005. Individual experience and experience working together: Predicting learning rates from knowing who knows what and knowing how to work together. *Management Sci.* **51**(6) 869–881.

Simon, H. A. 1955. A behavioral model of rational choice. *Quart. J. Econom.* **69**(1) 99–118.

Simon, H. A. 1996. *The Sciences of the Artificial*. MIT Press, Cambridge, MA.

Sproull, L., S. Kiesler. 1991. *Connections: New Ways of Working in the Networked Organization*. MIT Press, Cambridge, MA.

Stasser, G., D. D. Stewart, G. W. Wittenbaum. 1995. Expert roles and information exchange during discussion: The importance of knowing who knows what. *J. Experiment. Soc. Psych.* **31** 244–265.

Steiner, I. 1972. *Group Process and Productivity*. Academic Press, New York.

Teasley, S. D., L. A. Covi, M. S. Krishnan, J. S. Olson. 2002. Rapid software development through team collocation. *IEEE Trans. Software Engrg.* **28**(7) 671–683.

Thompson, J. 1967. *Organizations in Action*. McGraw-Hill, New York.

Van de Ven, A. H., L. A. Delbecq, R. J. Koenig. 1976. Determinants of coordination modes within organizations. *Amer. Sociol. Rev.* **41**(April) 322–338.

Waller, M. J. 1999. The timing of adaptive group responses to nonroutine events. *Acad. Management J.* **42**(2) 127–137.

Walz, D. B., J. J. Elam, B. Curtis. 1993. Inside a software design team: Knowledge acquisition, sharing, and integration. *Comm. ACM* **36**(10) 63–77.

Watson, D., T. Finholt. 1986. Measurement of group participation patterns. Working paper, Carnegie Mellon University, Pittsburgh, PA.

Wegner, D. 1995. A computer network model of human transactive memory. *Soc. Cognition* **13**(3) 319–339.

Weick, K., K. Roberts. 1993. Collective mind in organizations: Heedful interrelating on flight decks. *Admin. Sci. Quart.* **38**(3) 357–381.

Wood, R. E. 1986. Task complexity: Definition of the construct. *Organ. Behav. Human Decision Processes* **37**(1) 60–82.

Xia, W., G. Lee. 2005. Complexity of information systems development projects: Conceptualization and measurement development. *J. Management Inform. Systems* **22**(1) 45–83.