Team Knowledge and Coordination in Geographically Distributed Software Development

J. ALBERTO ESPINOSA, SANDRA A. SLAUGHTER,
ROBERT E. KRAUT, AND JAMES D. HERBSLEB

J. ALBERTO ESPINOSA is an Assistant Professor of Information Technology and UPS Scholar at the Kogod School of Business at American University. He received his Ph.D. in Information Systems from the Tepper School of Business at Carnegie Mellon University. His research focuses on understanding how teams coordinate across geographic and global boundaries and which team processes and information technologies are most effective in bridging these boundaries to achieve high levels of performance. His current research areas include global software and technical teams, team knowledge, team coordination, and spatial and temporal boundaries.

SANDRA A. SLAUGHTER is an Associate Professor in the Tepper School of Business at Carnegie Mellon University. Prior to joining the faculty at Carnegie Mellon University, Dr. Slaughter worked in industry as a project manager and systems analyst at Hewlett-Packard, Rockwell International, and Square D Corporation. She has consulted with the Information Technology Management Association and with several companies on software design and development-related issues. Her research is motivated by her experience in software development and focuses on the performance implications of software design and development decisions. Currently, she is conducting research funded by the National Science Foundation on project management practices and software design and evolution. She has also commenced new projects that explore capabilities and performance in information technology outsourcing.

ROBERT E. KRAUT is Herbert A. Simon Professor of Human–Computer Interaction at Carnegie Mellon University. He received his Ph.D. in Social Psychology from Yale University in 1973, and previously taught at the University of Pennsylvania and at Cornell University. He was a research scientist at AT&T Bell Laboratories and Bell Communications Research for 12 years. Dr. Kraut has broad interests in the design and social effect of computing and conducts research on everyday use of the Internet, technology and conversation, collaboration in small work groups, computing in organizations, and contributions to online communities.

JAMES D. HERBSLEB is an Associate Professor of Computer Science and the Director of the Software Industry Center at Carnegie Mellon University. He received his Ph.D. and J.D. from the University of Nebraska, and his M.S. in computer science from the University of Michigan, where he also completed a postdoctoral fellowship. His research focuses on coordination in software engineering, especially in globally distributed and open source software projects, as well as coordination in collaborative work more generally.

Journal of Management Information Systems / Summer 2007, Vol. 24, No. 1, pp. 135–169.

0742–1222 / 2007 $9.50 + 0.00.
DOI 10.2753/MIS0742-1222240104

ABSTRACT: Coordination is important in software development because it leads to benefits such as cost savings, shorter development cycles, and better-integrated products. Team cognition research suggests that members coordinate through team knowledge, but this perspective has only been investigated in real-time collocated tasks and we know little about which types of team knowledge best help coordination in the most geographically distributed software work. In this field study, we investigate the coordination needs of software teams, how team knowledge affects coordination, and how this effect is influenced by geographic dispersion. Our findings show that software teams have three distinct types of coordination needs—technical, temporal, and process—and that these needs vary with the members' role; geographic distance has a negative effect on coordination, but is mitigated by shared knowledge of the team and presence awareness; and shared task knowledge is more important for coordination among collocated members. We articulate propositions for future research in this area based on our analysis.

KEY WORDS AND PHRASES: coordination, global software development, management of the information technology (IT) function, team knowledge.

LARGE-SCALE SOFTWARE DEVELOPMENT REQUIRES a substantial amount of coordination because software work is carried out simultaneously by many individuals and teams, and then integrated into a single product. Software parts need to integrate and interoperate properly, and production schedules need to be synchronized, creating dependencies among tasks and people. These coordination challenges are compounded when the teams doing the work are distributed across multiple geographic locations. Developing software globally is increasingly becoming more attractive, in part because communication technologies have made it easy to communicate and exchange digital products across distances. Furthermore, global software development also affords wider geographical market coverage, closer proximity to clients, and better access to special software talent and technical resources [14]. On the other hand, when software is produced from multiple locations, it becomes more difficult to manage task dependencies and coordinate, increasing development time [38].

Despite today's sophisticated collaboration and software engineering tools, coordination continues to be challenging in software development. Many software projects are behind schedule and over budget, and do not always work as intended [53]. In part, these failures are due to problems with coordination [6, 21, 47], especially when software projects are large [9] and globally distributed [38]. Therefore, it is important to understand how geographic dispersion affects a team's ability to coordinate and deliver software in a timely manner. Most studies of team coordination in software teams have focused primarily on communication as the main mechanism teams use to communicate [18, 38, 41, 47, 51]. Moreover, the team cognition literature shows that real-time synchronous teams also coordinate through team knowledge about the task and about team members [13, 44, 56], but this perspective has not been adequately explored in more asynchronous and geographically distributed contexts.

Our goal in this study is to examine how geographically distributed software developers coordinate their work, with a particular interest in understanding whether and

how team knowledge helps them coordinate. Team cognition research suggests that as team members interact over time, they develop team knowledge that arguably can help them coordinate implicitly because they can anticipate what others are likely to do and can interact more effectively [12, 44, 45]. While individual knowledge is necessary in a software team, it is not sufficient. Large-scale software projects require the integration of knowledge from multiple technical and functional domains [21, 77]. Although there are reasons to think that team knowledge could improve coordination in software teams, it is not entirely clear how team knowledge may influence coordination in geographically distributed software development, nor is it evident how geographic dispersion may affect the effectiveness of various types of team knowledge on coordination. Therefore, our research addresses the following questions: *How do various types of team knowledge affect coordination in software development? And, how do these effects vary with geographic dispersion?*

We first identify the different types of coordination problems experienced in large-scale software development, based on the various types of dependencies developers face in their work and whether team knowledge helps software teams coordinate their work. We then explore how the effectiveness of various types of team knowledge is affected by geographic distance. The present study contributes to the information systems literature in several ways. First, although some studies have explored the effect of specific team knowledge mechanisms—for example, expertise coordination [29] or group mind [19]—on software team coordination, there are no studies that have jointly investigated the effects of various types of team knowledge on large-scale collaborative software development in a global context. Second, these studies have neither acknowledged nor distinguished between the different categories of team knowledge that may be used to coordinate implicitly, nor have they investigated how collocated and geographically distributed teams use these various types of team cognition. Finally, there are no prior studies on this topic carried out empirically, examining global software teams working in real software organizations.

Theoretical Foundations

Coordination in Software Development

COORDINATION THEORY DEFINES COORDINATION as the management of dependencies among task activities [52]. When the task activities of multiple individuals need to interrelate in a synchronized fashion, the corresponding interdependencies need to be well managed. Thus, a team can be said to be highly coordinated when its key task dependencies have been effectively managed (e.g., parts are well integrated into the finished product, task activities are completed on schedule, etc.). Complex tasks with tightly coupled dependencies such as collaborative software development can benefit from coordination [21, 47]. For example, each developer on a team may individually produce software code effectively (e.g., error free, on time, and of high quality), but this new software may not work well with software parts produced by other developers.

From a technical perspective, the integration of multiple software parts needs to be well coordinated through mechanisms such as team communication, specifications,

and configuration management. However, this type of technical dependency is not the only coordination challenge faced by software teams. From a project management perspective, developers also need to complete their tasks according to an established schedule. Temporal coordination is particularly important for the performance of global project teams [55]. Slippages in the schedule in some parts of the code or software activity can put the whole project behind schedule. For example, if the design of a particular feature is a few days late and the development of this feature has already been scheduled, the development team will need to stand by until the design is finished or wait to be reassigned to other activities. Similarly, from a software process perspective, improper management or lack of compliance with the established software processes can have negative effects on software project outcomes [22] and can create problems or delays because of things such as "features churn" [19], priority conflicts, and blocking issues. Given these various types of dependencies present in software projects, we anticipate that the sensitivity to the respective types of coordination problems will vary depending on the role of the person involved (e.g., technical, manager) and the types of dependencies that affect their work the most. This is an important aspect of our study because most prior studies of coordination have relied on scales that do not distinguish among these various types of dependencies and roles.

Coordination Through Team Knowledge

The classic organizational literature indicates that team members coordinate "mechanistically" via "task programming mechanisms" and "organically" through team communication [54, 73, 75]. Adopting the right mix of coordination mechanisms can make software teams more productive [4], which is critical in competitive markets [40]. Coordination of repetitive and routine aspects of the task can be achieved mechanistically using tools, schedules, plans, manuals, and specifications. A configuration management system, which enables developers to work simultaneously in different parts of the code without interfering with each other, is an example of a task programming mechanism. More uncertain aspects of the task (e.g., missed deadlines or hardware failures) cannot be effectively coordinated mechanistically, so members need to coordinate "by feedback" or "organically" through communication [54]. While these traditional coordination mechanisms are important, other cognitive factors can also influence how teams coordinate. Team members' common ground, knowledge of each other and the task domain, familiarity with task programming mechanisms (e.g., software tools, specifications), and awareness of who is around and who has done what recently are a few examples of these factors.

The literature on team cognition suggests that as team members interact with each other and gain expertise with the joint task, they develop team knowledge about the task and the team, which helps them to coordinate implicitly [12, 44]. Such implicit coordination has been referred to as the "synchronization of member actions based on unspoken assumptions about what others in the group are likely to do" [84, p. 3]. However, how these cognitive mechanisms may help in asynchronous and geographically distributed software tasks has not been effectively addressed in the literature.

Several streams of research have defined and studied different types of team knowledge mechanisms, including team mental models [13], shared schemas [65], collective mind [82], transactive memory [81], and team situation awareness [24]. Although these constructs are conceptually different, they are all based on some form of team knowledge. Cooke et al. [17] elaborated the concept of team knowledge to help us reconcile these multiple views, suggesting that team knowledge can be classified into two general categories—long-term knowledge and fleeting knowledge or "awareness."

This classification is useful when studying team cognition and coordination because both types of knowledge can help teams coordinate their work. Long-term knowledge is acquired over time, is more permanent, and has applicability over several phases of the task. This is the type of knowledge acquired through training, education, and experience, and it has relevance at various stages of the task. For example, a developer's knowledge of software engineering will be useful to that individual's software development efforts throughout the task. Long-term knowledge can either be individual (needed to carry out individual tasks) or shared (needed to understand each other's tasks and coordinate the work).

In contrast, fleeting knowledge or "awareness" changes depending on the specific task situation. Awareness is situational and it is only relevant until the situation changes. For example, a developer's knowledge of who is responsible for testing a particular software file and whether this testing has been completed is only useful if the developer is waiting for that particular file to be tested. A few weeks later, this knowledge may no longer be useful. Awareness can also be individual—needed to understand one's task environment—and team—needed to know what others are doing to synchronize their actions. Team awareness enables members to have up-to-the-minute information about what is happening with the shared task environment and the team, thus helping them make adjustments to stay synchronized with other team members. In this study, we examine both long-term shared knowledge and team awareness. Specifically, we explore how coordination in global software development is influenced by two types of shared knowledge—shared knowledge of the task and of the team—and two types of team awareness—task awareness and presence awareness—which we discuss more thoroughly below.

Shared Knowledge

Shared knowledge develops over time from prior familiarity with common tools and processes, the product being developed, the task domain, and team members, and is long-lasting. This type of knowledge may exist prior to the current task and will further develop during the task. Knowledge of who knows what in the team and knowledge of the process used by a team to develop a particular software application are examples of shared knowledge. Shared knowledge helps members coordinate because it helps individuals develop more accurate explanations and expectations about task events and member behaviors [12, 44, 45, 66]. As Alavi and Leidner [2] concluded in their literature review of knowledge management research, shared knowledge is important because knowledge is processed in people's minds and, therefore, individuals need to

have a certain level of overlap in their individual knowledge bases to coordinate their collective action. Furthermore, shared knowledge, particularly when members know that they share it (i.e., "mutual knowledge"), also helps team members coordinate because their communication is more efficient due to shared vocabularies and more common ground [18, 31, 46].

Team cognition research suggests that team members develop organized shared knowledge about many things (e.g., goals, strategies, processes, team interaction, etc.), but that the knowledge that matters the most for task performance relates to either task work (i.e., activities necessary to carry out the task) or teamwork (i.e., activities necessary to work with each other) [17, 44, 64]. Having shared knowledge about technical concepts, products, and processes can help software team members develop accurate expectations about future states of the task and improve common grounding in their communication, which helps coordination. For example, a study of software teams found that one of the most salient problems leading to mistakes and the need for additional effort was the thin spread of application domain knowledge within the team [21]. This study and others (e.g., [77]) have concluded that task knowledge sharing and integration is necessary to ensure positive outcomes in this domain. Consequently, we anticipate that shared knowledge of the task can be beneficial for team coordination in global software development.

Similarly, having knowledge about members of the team could also help coordination because individuals can develop accurate perceptions of what other teammates know and how they may respond to particular events and circumstances, thus helping them plan their own actions. For example, transactive memory research suggests that knowledge of who knows what in the team helps their coordination because members know who to contact when they need information, and also because members develop expectations about who in the team will pay attention, acquire, and process what kinds of new information when it arrives to the team [8, 50, 80, 81]. The beneficial effects of knowing who has which expertise in the team have been observed with consulting teams made up of MBA students [49] and large-scale software teams [29]. Other studies with software teams have also demonstrated the importance of integrating individually held expertise [74] and suggested that understanding how one's work contributes to other team members' tasks helps develop a collective mind [82], which helps team members become more coordinated [19]. While the constructs investigated in these studies are conceptually different, these studies collectively share the underlying notion that having knowledge about team members is beneficial for coordination. Therefore, we anticipate that shared knowledge of the team has the potential to facilitate team coordination in global software development.

Team Awareness

Team awareness is knowledge about what is happening in the team's task environment at any given point in time and is fleeting. This is related to the concept of situation awareness, which is up-to-the-minute perception and comprehension of what is happening in the task environment, and an understanding of how this will affect the

task [1, 24], which has been studied extensively in aviation and other real-time tasks. Knowledge of an upcoming deliverable deadline and knowledge of the progress of the development of a particular software module are examples of team awareness, which helps members synchronize their actions with other team members [24, 83]. Team awareness has been defined simply as "an understanding of the activities of others, which provides a context for your own activity" [23, p. 107]. Gutwin and Greenberg [36] concluded from their several studies on the subject that team awareness is important for coordination in collaborative tasks that contain interdependent activities, because it helps members shift from individual to shared activities seamlessly and easily, and because members have a better understanding of the sequence and timing of things and the temporal boundaries of their actions. There are several types of team awareness that have been discussed in the literature, including work space awareness, activity awareness, environmental awareness, and task awareness and presence awareness, which are particularly popular among collaboration tool design researchers [25, 34, 69]. In this study, we focus on two types of team awareness that are important for coordination—"task awareness" and "presence awareness." We selected these two types of awareness for our study because they are most important in virtual collaboration [69] and because they provide situational knowledge about task work and teamwork.

Consistent with the definitions of team awareness discussed above, we define task awareness as a member's up-to-the-minute knowledge of what is going on in the task in areas that affect that member's work. This definition is similar in concept to Chen and Gaines's [15] concept of chronological awareness, which is knowledge of recent task activities (e.g., knowing who did what recently, who is behind schedule, what tasks are pending). Chen and Gaines argue that this type of awareness provides essential information for collaborators engaged in a task that is too large or too complex for a single member to undertake. This implies that knowing the task activities of other teammates could help team members to coordinate their work more effectively in global software development tasks.

Presence awareness—up-to-the-minute knowledge of which team members are around, where and when, as relevant for the task—is also important when members collaborate on a task. Presence awareness has been investigated by computer scientists and software development researchers [7, 33, 37] because of its potential to bring some sense of collocation to geographically distributed teams [30]. But most efforts in presence awareness research have focused on tool design. In fact, presence awareness tools and features are becoming very popular in corporate collaboration applications [58] because of their potential benefits. When members have tight dependencies with other members, it is important to be able to find the right people when you need them, or at least to know when and if they are around. This is generally not a problem in collocated environments where members have abundant presence awareness cues (e.g., coat hanging in the closet, car parked in the lot, office light is on), but knowing where people are can be a challenge in geographically distributed environments where such presence cues are not generally available. Therefore, presence awareness could be helpful in facilitating coordination in global software development tasks.

Team Knowledge in Geographically Distributed Collaboration

Most of the theory building and empirical research on team cognition has concentrated on real-time and collocated contexts [56]. However, some of the theories and findings in this area may not necessarily extend to geographically distributed contexts [11]. Thus, it is imperative that we develop an understanding of the fundamental differences among these contexts before we can make generalizations beyond real-time and collocated contexts. Geographic dispersion affects the nature of interaction within the team and provides fewer opportunities for spontaneous interaction and acquisition of team knowledge [3]. Because of this, it often takes longer for members to get an acknowledgment, obtain an answer, or correct miscommunication. Furthermore, studies have shown that a substantial amount of coordination in software development takes place through informal encounters and meetings in public places such as the water cooler or coffee room [47, 63], which does not happen when members are separated by distance. Indeed, a recent empirical study examined the "radical collocation" of software development teams and found significant benefits of collocation in terms of facilitating coordination, learning, and performance [72]. Consequently, we anticipate that geographic dispersion hinders coordination in software development [21, 38, 39].

While this may not seem like a novel prediction, it is important to point out that the effects of geographic distance on coordination have not been conclusively determined, mainly because geographic distance often correlates with other boundaries that affect coordination in global teams such as time zones, cultural differences, and technology mediation [26, 61, 78]. In an article summarizing their several years of research on the effects of distance in teams, Olson and Olson [59] concluded that many of the observed effects of distance are due to factors other than distance per se. Similarly, more recent studies have shown that factors such as cultural differences and time zones affect [28] coordination more strongly than geographic distance alone. In contrast, recent studies of global teams suggest that distance per se is still a substantial barrier in collaborative work [42]. We conclude from our analysis of the literature that the issue of whether physical separation affects coordination has not been resolved. Nevertheless, we speculate that until technologies that can bridge distance become more mature and effective, distance will continue to be a challenge for many organizations.

We argue that geographic dispersion affects how teams coordinate in two fundamental ways. First, some mechanisms available to collocated teams are not available to geographically distributed teams. For example, collocated teams can communicate frequently and spontaneously, which is not the case for geographically distributed teams. Second, the effectiveness of those mechanisms that remain available despite member dispersion is affected by lack of copresence. For example, while geographically distributed teams can communicate in real time through video conferencing, some contextual references are lost when communicating through this technology. Consequently, the mix of coordination mechanisms used by collocated and distributed teams will differ. Generally speaking, coordination by communication will be more affected by geographic separation than task programming mechanisms because

distance impairs the team members' ability to communicate effectively, frequently, and spontaneously. Therefore, we expect that team cognition is particularly useful in helping geographically distributed team members coordinate because it helps them offset these communication barriers.

For example, shared knowledge of the task will afford more common ground among team members because of shared vocabulary and familiarity with technical terms, making their limited communication more effective [18]. Similarly, shared knowledge of the team will make it easier for its members to locate experts when needed, thus helping coordinate the access to and utilization of that expertise [29]. Task awareness also makes the team's limited communication more efficient because, unlike collocated team members, geographically distributed members can have long periods in which they do not know what their cross-site teammates are working on [69]. Finally, presence awareness, when available, can bring some of the benefits of copresence to distributed environments promoting more frequent spontaneous interaction among team members [30, 69], which research has shown to help software teams coordinate [47, 63].

Generally speaking, any mechanism that can help teams cope—for example, communicate more efficiently or reduce the need to communicate, anticipate and explain events and member actions, learn who knows what or where to find expertise, figure out who did what and when on particular task activities, and who is around—has the potential to offset some of the coordination problems. We expect this to be true for both collocated and geographically distributed collaboration. At the same time, because the ability to communicate effectively changes with distance separation [3], we anticipate that collocated and distributed team members will differ in terms of which team cognition mechanisms they use and how effective these mechanisms are in helping them coordinate. In the present field study, we try to fill this gap in the literature by developing further insights into how distributed software development teams coordinate their work, how various types of team knowledge—shared knowledge of the task, shared knowledge of the team, task awareness, and presence awareness—influence coordination in software development, and how these effects are affected by geographic dispersion.

Research Methodology

Study Context

WE STUDIED COORDINATION IN SOFTWARE TEAMS at a major division of a large telecommunications firm that developed software for wireless GSM (Global System for Mobile Communications) networks in Europe. We selected this firm for our study because of its extensive use of global teams for software development. Software teams for wireless networks, such as GSM, present an ideal context for this study because the respective software developers, technical managers, and project managers, as well as their customers, are distributed around the globe, and because GSM software systems are large and complex, developed incrementally over time. The interdependent nature of GSM software development makes it ideally suited for coordination studies.

A GSM network is composed of various network elements (e.g., base station controller, speech transcoder frame, base transceiver station, mobile service switching center, and operations and maintenance center). Each network element is a system of hardware and software that is periodically updated through product releases. A product release includes a bundle of new or improved features to the existing wireless network and generally incorporates releases for all or most of its network elements. Software development for different network elements is carried out by separate internal organizations often located in more than one geographic site. A software release for a given network element is generally assigned to a "release team" that implements the various features associated with that release.

We selected the release team member as the unit of analysis for this study for a number of reasons. First, release teams are relatively large (i.e., 50 or more developers), and large teams provide a good sample of participants and a wide variety of perspectives and stories about the software development process. Teams of this size are not uncommon in organizations working on complex product development releases or software versions where the work is subdivided by specific modifications and feature implementations assigned to smaller groups within the team. Second, these teams are project driven and very focused, and have a stable membership during the release implementation period (i.e., approximately one year), such that team members have the potential to develop a sense of identity with their team despite their size. Finally, these teams often involve developers from more than one geographic location, so comparisons can be made between collocated and cross-site collaborations.

Method

We conducted individual, semistructured face-to-face interviews with team members working on a GSM software release. Each interview was focused on one particular software modification or new feature implementation for the release that the participant recalled as salient, which we then used to elicit more specific instances of coordination successes and failures. We employed this research method because in-depth interviews help the researcher to acquire a richer understanding of the phenomenon under study at the early stages of an investigation and can thus inform subsequent stages of the research study when developing survey scales and variables [71]. We were particularly interested in understanding the nature of the dependencies in the global software development domain to learn how team cognition mechanisms help teams coordinate and how this coordination differs between collocated and geographically distributed team members.

Sample

The sample for this study included individuals on a large software team, including 36 software developers, technical managers, and project managers[1] located in England (n = 15) and Germany (n = 21). These individuals used mostly voice communication (telephone, teleconferencing, etc.), technical Web sites, and a configuration

management system[2] to coordinate their work. Like most organizations developing large-scale software, this organization used a configuration management system. This system helped developers to work simultaneously on the software code without affecting each other, manage software changes and versions, and document important software modification issues. We conducted interviews until we reached theoretical saturation—that is, the last few interviews did not provide new insights—making the sample size appropriate for this study [70]. We selected this development team because it had completed more than 70 percent of the software code at the time of the study, so its members could recall recent experiences relevant for the interview. Also, although developers on this team sometimes collaborated with colleagues from other countries, most of the software development for this release was carried out from these two locations. In addition, developers in these two locations had collaborated over the past few years. Finally, all German software developers on the team were fluent in English, and both sites were separated by only one time zone, thus reducing possible confounds stemming from differences in time zones, language, and other team boundaries [26], and ensuring that the effects observed were primarily due to geographic distance. We interviewed all software developers and managers from this release team who were available in these two locations at the time of the site visits, representing approximately 72 percent of the personnel assigned to this network element release.

Data Collection and Coding

Our data collection and coding methods are largely based on grounded theory [32, 70]. Grounded theory is a widely used qualitative method in information systems research [10, 60, 76] and global teams [62], particularly when the study is exploratory and the theoretical development of the topic is in its early stages [60]. Our method differs slightly from grounded theory in that we first conducted all our interviews and then analyzed the data. Grounded theory recommends doing data collection and analysis simultaneously, redirecting the inquiry based on what the emerging data suggest, but we decided to conduct several interviews up-front for practical considerations because we had a limited amount of on-site observation time allowed by the group. Our questions focused on a specific problem that the participant identified at the beginning of the interview, and our target participant group was initially well defined, so we did not need to alter our interview instrument and protocol.

All interviews except two were audiotaped. In one interview, the participant did not agree to be audiotaped, and in another interview, the tape recorder was not working. Substantial written notes were taken for these two unrecorded interviews. Interviews were limited to one hour, as requested by team managers. Most of the questions (see Appendix A) used were framed to uncover how team members managed their dependencies [52] when coordinating. The interviews were semistructured to allow participants to discuss or elaborate on important issues they recalled, even if unrelated to the specific question or incident they were originally answering. Participants were first asked a few background questions and were then asked to think of an important recent modification request or feature implementation for the network element release under

study that was salient in their minds. All subsequent questions were made in reference to that particular modification request or feature. The intention was to obtain specific information about whom they communicated with, which types of information they exchanged, which types of coordination challenges or failures they encountered, and which types of team cognition, if any, were used to coordinate tasks while developing that particular feature or modification request.

The data from the interviews were transcribed into a text document. The text contained approximately 480 single-spaced pages of written material. As prescribed by grounded theory, we first did open coding of the text data. Open coding focuses on uncovering general recurrent themes. We then did axial coding of the data, which involves finding relationships among these themes, which we then used to produce a template with hierarchical codes. Hierarchical coding schemes are not only useful because they allow fine-grained detail to be captured but findings can then be aggregated to higher levels to make generalizations. In order to keep our inquiry consistent with our research goals, we established a hierarchical coding scheme starting with three high-order code categories we defined based on our interest. All subsequent code subcategories underneath these three main categories emerged from axial coding of the data.

We defined the first high-order code category to uncover attributions [68] made by participants about the effect of different types of team knowledge on coordination. Attributions were made when a participant indicated that a particular type of team knowledge was important for coordination, or that its absence was detrimental to coordination. The second high-order code category we defined was to identify recurring themes related to specific work contexts—collocated (e.g., staff overload, getting people's attention) and cross-site work (e.g., little opportunity for interaction, low richness of communication media). The third high-order code category we used was to classify instances of coordination problems mentioned into more general types of coordination problems—technical, temporal, and process.

We only coded 32 of the 36 cases because four of the cases were from special interviews with product managers and cross-network element coordinators who were not directly involved in the development effort of the release we studied but were very knowledgeable about coordination issues related to the integration of features across different network elements because of their intimate involvement with client requirements. We used the content of the four interviews to complement our interpretation of the 32 main interviews—10 managers and 22 technical staff. The coding scheme that emerged after axial coding is shown in Appendix B. This method is similar in concept to codebook analysis or thematic coding [43] and content analysis [79], except that the codebook is not established before the study. Rather, it emerges from investigation of recurrent themes, and the data are analyzed not only statistically but also through qualitative interpretation [43]. Coding of textual data enables researchers to classify text segments into meaningful information that can be retrieved for interpretation during analysis [57].

Appendix C shows an example of a coded case. After all the interviews were coded by one of the researchers, we asked an independent coder to code six randomly selected

interviews using the same coding template. We then analyzed and discussed agreements, disagreements, and coding errors on these six interviews with the independent coder. This helped the external coder become more familiar with the coding scheme. We then asked the independent coder to recode these six interviews and also code the remaining interviews. A comparison of the episodes coded by the researcher and the independent coder on all 32 interviews yielded a final agreement rate of 76.6 percent, measured as number of agreements over agreements plus disagreements [57], and a kappa reliability of 72.1 percent [16], indicating substantial agreement between the coders [48]. The final coding of disagreements was discussed and resolved jointly with the independent coder after inspecting each disagreement.

The cases were analyzed first by sorting text segments by their respective codes to uncover patterns of responses for each code in the coding scheme. Consistent with grounded theory, we further analyzed the data by evaluating similarities and differences in the cases. The primary comparison we made was between collocated and geographically distributed contexts, but we also compared similarities and differences among different types of cognition—shared knowledge of the task, shared knowledge of the team, task awareness, and presence awareness—and coordination problems—technical, temporal, and process—discussed. In addition, we also counted the number of cases in which specific attributions were made. Even though we did not rely on counts for our analysis, we used counts or frequencies to assess trends and the prevalence of recurrent themes. Counts are useful because they help direct attention to aspects of the data that warrant further investigation [43], evaluate interrater reliability of the coded data [68], and reduce problems with analytical bias [57]. However, counts do not tell anything meaningful about the data without qualitative interpretation of the narrative recorded from participant responses [43, 57]. Thus, as recommended for qualitative research [32, 57, 70], we analyzed our data through interpretation of similarities and differences in the coded text.

Analysis and Results

WE FIRST IDENTIFIED AND ANALYZED THE TYPES of coordination problems that were salient to our study participants because we were interested in developing a deeper understanding of our dependent variable. We then examined how geographic dispersion influenced coordination, based upon the comments of the interviewees. Finally, we analyzed the role of different types of team cognition in facilitating coordination in the global software development projects described by team members, and how the use of team knowledge was affected by geographic dispersion. Throughout, we develop and present propositions based upon our findings.

Coordination Types

Miles and Huberman [57] suggest organizing and displaying results from textual data in a matrix form. We employed the "code category by role" matrix shown in Table 1 to present our coding results for the number and percentage of cases in which each

Table 1. Coordination Problems Discussed

Code category	Response coded	Role					
		All		Managers		Technical	
		Participants	Percent	Participants	Percent	Participants	Percent
Coordination types described	Technical	28	88	6	60	22	100
	Temporal	24	75	10	100	14	64
	Process	18	56	9	90	9	41

Notes: Numbers show how many participants discussed these types of coordination problems. Percentages are based on the number of participants.

coordination problem type was discussed by a technical staff or manager. Our analysis suggests that there are different types of coordination problems in global software development, and that these problems depend on the type of dependencies involved. The different coordination problems discussed by participants were categorized into one of three general types of coordination they identified—technical, temporal, and process. Technical coordination problems were discussed by 28 (88 percent) of the participants. These problems surfaced when technical dependencies among software parts were not effectively managed (e.g., redundant code, incompatible interfaces, integration problems). Temporal coordination problems were discussed by 24 (75 percent) of the participants. These problems occurred when time dependencies were not effectively managed, such as when software parts or software activities were not finished according to project schedules, affecting the work of others (such as when testing cannot start because coding is not complete). Finally, process coordination problems were brought up by 18 (56 percent) of the participants. These problems surfaced when dependencies in the software development process were not effectively managed (e.g., nonadherence to the established software process, priority conflicts, development work starts before its design is certified, etc.). The following are examples from the interviews of these three types of coordination problems:

> Technical coordination: "Parts for two different network elements worked well individually, but not together because the specs were misinterpreted, which delayed the testing by six weeks." (Testing Engineer)
>
> Temporal coordination: "Being last in the process comes with the territory. . . . If things aren't ready on schedule I need to replan all my team's work. On average, schedules are one or two days late." (Testing Engineer)
>
> Process coordination: "There are misunderstandings, which complicate things . . . in one instance, the Gate 3 [i.e., end design-start coding] review took place after the coding was completed. You should not start coding until the Gate 3 review is completed. . . . We are still waiting to hear which features are in and which ones are out." (Technical Manager)

Based upon these findings, we propose:

> *Proposition 1: The coordination problems experienced by software development teams fall into one of three categories that correspond to the types of dependencies they need to manage—technical, temporal, and process.*

It is important to note that we are not positing that these different coordination types are collectively exhaustive or orthogonal. In fact, we suspect that these types of coordination may be interrelated in some way and that there may be other types of coordination that did not surface in our interview. For example, team members may have a pooled dependency [73] on scarce shared resources such as specialists and dedicated hardware. Similarly, technical coordination problems or priority conflicts (i.e., process coordination problems) may lead to rework and delays, which may create temporal coordination problems. Nevertheless, these three types of coordination

problems were salient to our participants and distinguishing among them is important because it helps identify the different types of dependencies present in a collaborative task and how these dependencies affect different groups.

Interestingly, as shown in Table 1, participants in different roles had different perspectives about coordination problems, depending on the type of dependency that affected them the most ($\chi^2 = 21.6$, degrees of freedom [df] = 2, $p < 0.001$). All of the technical staff (100 percent) discussed at least one type of technical problem, but only six (60 percent) of the managers discussed technical coordination issues. In contrast, all managers (100 percent) discussed at least one type of temporal coordination problem and nine of them (90 percent) mentioned process coordination problems, while only 14 (64 percent) and nine (41 percent) of the technical staff mentioned temporal and process coordination problems, respectively. These results suggest that software professionals in different roles are more sensitive to the particular types of coordination that more directly affect their work.

Technical groups design, develop, and test software parts that need to interoperate properly when integrated, so it is not surprising that they were more concerned with managing technical dependencies. Although many technical staff discussed temporal and process coordination problems, these problems were not as salient to them as technical coordination problems. In contrast, managers were more concerned about managing project schedules and the software development process. Consequently, managers were more sensitive to the need to manage temporal and software process dependencies (i.e., temporal and process coordination). While many managers discussed technical coordination problems, such problems were not as salient to them. This leads us to propose that:

> *Proposition 2: The types of coordination problems faced by software development collaborators vary by role, depending on the nature of the dependencies that affect their individual work—that is, team members in technical roles experience more technical coordination problems, whereas team members in management roles experience more temporal and process coordination problems.*

Effects of Geographic Dispersion

Table 2 presents our coding results for the number and percentage of cases in which each coordination issue was discussed by collocated team members. We did not find substantial differences in the relative importance of these coordination problem types between collocated and geographically distributed contexts. In other words, in both contexts, technical coordination problems were more salient than temporal coordination problems, which were, in turn, more salient than process coordination problems. However, our analysis of the interviews suggests that coordination is generally less problematic with collocated developers than with those collaborating across geographical locations. Because we took special care in our research design to mitigate confounding effects with other team boundaries such as language and time zones, this represents an important finding and validates prior findings that geographic

Table 2. Results on Salient Issues by Context

			All	
Code category	Collaboration context	Response coded	Number of participants	Percent
Context-specific issues	Collocated	Not a lot of coordination problems collocated	17	53
		Priority conflicts are a problem	8	25
		Overload is a problem	15	47
		Getting people's time/attention is a problem	11	34
	Across sites	Little opportunity for informal interaction	5	16
		Lean communication media (low richness)	11	34
		Other: time zones, language, culture, and so on	13	41
		Delays more substantial across sites	7	22
		Prior knowledge of colleagues/context helps	22	69
		Need redundant or liaison roles (they help)	12	38

Notes: Numbers show how many participants discussed these issues in the respective context. Percentages are based on the number of participants.

dispersion is a substantial barrier to getting the work done [20, 42]. A majority of the participants (17, or 53 percent) explicitly mentioned that they did not have many problems coordinating their work with collocated colleagues because they knew who they were and where to find them, and because they often encountered each other in public areas such as hallways and lunchrooms, which enabled them to have frequent, rich, and spontaneous interactions. In contrast, the majority of participants (29, or 91 percent) mentioned at least one problem relating to geographically distributed work, and many of them (22, or 69 percent) mentioned at least one type of problem that was specific to geographically distributed work (e.g., fewer opportunities for interaction, no presence awareness).

Our findings shown in Table 2 also suggest that geographic distance influences the effectiveness of organic coordination through communication because of different reasons, including lack of familiarity with cross-site colleagues and working environments (22, or 69 percent), lean communication media (11, or 34 percent), and fewer opportunities for interaction (5 or 16 percent). The following comment by a developer illustrates how distance affects coordination:

> "Sometimes I don't know how remote people fit into specific groups. I only know two or three people [at the other site], and the rest of the organization [at the other site] is a black box. Locally, it is easier to change the team configuration when things are not working well, but this is not possible with remote collaborations." (Developer)

This suggests the following proposition:

> *Proposition 3: Geographic distance hinders team coordination through communication in global software development.*

Effect of Team Knowledge on Coordination Success

Table 3 summarizes the coding results with the number and percentage of cases in which each attribution about team knowledge was made in the context of collocated or geographically distributed collaboration. About 91 percent of all participants mentioned at least one type of team knowledge being important for coordination. Although it may seem intuitively obvious to suggest that shared knowledge helps teams coordinate, this effect has been explored primarily in synchronous and collocated collaboration contexts and less so in asynchronous and distributed contexts. Furthermore, we know little about how these effects differ across collaboration contexts. Consequently, we start with simple general propositions about the effects of shared knowledge and team awareness on team coordination, and in the next subsection we discuss how these effects differ across contexts.

With respect to shared knowledge, many participants expressed comments suggesting that both shared knowledge of the task (24, or 75 percent) and shared knowledge of the team (25, or 78 percent) were important when coordinating their work. This is illustrated by the following comments:

Table 3. Results on the Importance of Team Knowledge for Coordination Effectiveness

		Collaboration context					
		All		Collocated		Across sites	
Code category	Response coded	Number of participants	Percent	Number of participants	Percent	Number of participants	Percent
Importance of type of team knowledge for coordination	Shared knowledge of the task	24	75	20	63	10	31
	Shared knowledge of the team	25	78	4	13	25	78
	Task awareness	9	28	6	19	6	19
	Presence awareness	12	38	1	3	12	38

Notes: Numbers show how many participants discussed these attributions. Percentages are based on the number of participants.

> Shared knowledge in general—"understanding the technology that others refer to is a major problem, especially with systems architects. . . . The difference in common language is sometimes technical and sometimes conceptual. . . . Currently I have to spend a lot of time explaining things because many new colleagues have no knowledge of our product. . . . Sometimes I don't know how to interact with them." (Developer)
>
> Shared knowledge of the task—"we simply increase the number of status meetings [to address coordination issues] so that everyone is aware of everyone's needs. I think we now understand our problems better. Therefore, we are able to build contingencies and more realistic schedules." (Testing Engineer)
>
> Shared knowledge of the team—"new teams need time to coagulate [i.e., gel] [to coordinate effectively], get to know each other, know who knows what, know who to trust, know how to work together, etc. . . . need less people churn and more time to get to know each other." (Testing Engineer)

Thus, we posit that:

> *Proposition 4: (a) Shared knowledge of the task and (b) shared knowledge of the team help team coordination in software development.*

Half of the people we interviewed (16, or 50 percent) discussed the importance of at least one type of team awareness for coordination and most of these comments were made in relation to distributed collaboration (14, or 44 percent). Some participants suggested the importance of task awareness (9, or 28 percent), whereas others discussed the importance of presence awareness (12, or 38 percent). Problems with task awareness included things such as not knowing whether a particular software part was ready for testing or whether a particular design was ready to start coding. While task awareness is important in any collaborative effort, it is particularly important in geographically distributed contexts because it is more difficult to figure out who has done what with respect to the task, as the following comment made by a testing engineer illustrates:

> "[We need to exchange] information about MRs [i.e., modification requests] raised as a result of problems found so that they [i.e., developers] can start fixing. Also, [we need to exchange information] about which tests we plan to do and which ones we have [already] done."

Problems with presence awareness have to do with not knowing when people are around. The following comment made by a developer illustrates the importance of presence awareness for coordination:

> "It is hard to know people's availability. Sometimes you learn very late that a developer is no longer around and you are wondering why you did not get an e-mail reply. Sometimes it takes a while to get replies and it is because people are not available."

Thus, we posit that:

Proposition 5: (a) Task awareness and (b) presence awareness help team coordination in software development.

Differences in Team Knowledge Effects on Coordination: Collocated Versus Distributed Work

While team knowledge appears to be universally beneficial for coordination, interestingly, there are differences in the importance of the specific types of team knowledge on coordination between collocated and geographically distributed work. As Table 3 shows, we found significant differences between collocated and geographically distributed team members in terms of the importance of different types of team knowledge for coordination ($\chi^2 = 23.7$, df = 3, $p < 0.001$). Except for task awareness, participants discussed different types of team knowledge being important for collocated and distributed work. Participants mentioned the importance of shared knowledge of the task for coordination more often in the collocated context (20, or 63 percent) than in the geographically distributed context (10, or 31 percent), while they discussed shared knowledge of the team (25, or 78 percent) and presence awareness (12, or 38 percent) more often in the geographically distributed context than in the collocated context (4, or 13 percent, and 1, or 3 percent, respectively). This result suggests that software developers were more concerned about having shared knowledge of key products, concepts, and processes with their collocated collaborators, but they were more concerned about knowing the skills, expertise, and abilities of their colleagues at other sites and being able to find them when needed. The majority of participants (22, or 69 percent) indicated that having prior knowledge of colleagues or of the work environment on the other site offsets many of the problems of working with remote colleagues, as the following comment illustrates:

> "I don't have a lot of problems [across sites] because I know the team in Germany well. . . . I spent nine months training there. . . . I know who to talk to when I have a problem." (Testing Engineer)

While shared knowledge of the task may be important for geographically distributed work, it is not entirely surprising that participants did not discuss it much during our interviews given that many of them had difficulties knowing their colleagues at other sites. We speculate that if team members do not know who is who, the lack of shared task knowledge will not be very salient to them. If team members do not know their colleagues at other sites, it is not possible for them to know whether they have any task knowledge in common. Conversely, collocated team members who know each other well are more sensitive to problems stemming from the lack of common understanding and the lack of shared beliefs about concepts, products, and processes (i.e., shared mental model of the task). The importance of knowing colleagues at other sites is illustrated by the following comment:

> "We need to have early face-to-face meetings to get to know people we work with at other sites to figure out who to rely on and who to approach for a given problem." (Technical Manager)

This leads us to propose that:

Proposition 6: The most effective mix of team knowledge mechanisms for coordination will vary depending on whether the work is carried out with collocated or geographically distributed members.

Proposition 7: (a) Shared knowledge of the task is more important for collocated members and (b) shared knowledge of the team is more important for geographically distributed team members.

Proposition 8: Shared knowledge of the team helps offset some of the negative effects of geographic dispersion on coordination.

We did not find any difference in the importance of task awareness for coordination between collocated and geographically distributed work at this organization. One possible explanation for this finding is that software developers of large systems such as the one we studied use sophisticated software collaboration tools such as configuration management systems that provide most of the task awareness that is needed to do their jobs. Empirical studies have found that these systems help teams coordinate the technical aspects of software tasks [27, 35]. These systems track who has changed which parts of the code, thus providing a substantial amount of task awareness to team members. These systems also protect parts of the code automatically, so that developers can make simultaneous changes in the same software code without affecting each other, thus making task awareness less necessary. Thus, we do not have empirical evidence about the importance of task awareness, but we speculate that task awareness is critical to performance in both collocated and distributed collaboration, and that task awareness can be effectively provided with sophisticated collaboration tools such as configuration management systems and up-to-date technical and project Web sites.

In contrast, we found substantial differences in the importance of presence awareness for coordination between collocated and geographically distributed work. Presence awareness problems across sites included things such as not knowing when colleagues were at their desks, when they were on holiday or vacation schedules, or their general whereabouts. Problems with lack of presence awareness in geographically distributed collaboration have been identified in prior internal studies at this organization, which led to the development of a number of collaboration tools, including a shared team calendar; a team presence awareness tool with team chat features; and a team portal tool with information about national holidays, time zones, and other important information about different sites in which a team operates [5]. These tools were being deployed and evaluated at the time of this study. This leads us to propose that:

Proposition 9: Presence awareness is more important for coordination with geographically distributed team members than with collocated members.

Proposition 10: Presence awareness can offset some of the negative effects of geographic dispersion on coordination.

Table 4 provides an overall summary of our results and propositions.

Table 4. Results and Propositions Summary

Issue	General effects	Dispersion effects
Coordination	P1: The coordination problems experienced by software development teams fall into one of three categories that correspond to the types of dependencies they need to manage—technical, temporal, and process.	P3: Geographic distance hinders team coordination through communication in global software development.
	P2: The types of coordination problems faced by software development collaborators vary by role, depending on the nature of the dependencies that affect their individual work—that is, team members in technical roles experience more technical coordination problems, whereas team members in management roles experience more temporal and process coordination problems.	P6: The most effective mix of team knowledge mechanisms for coordination will vary depending on whether the work is carried out with collocated or geographically distributed members.
Shared knowledge of the task	P4a: Shared knowledge of the task helps team coordination in software development.	P7a: Shared knowledge of the task is more important for collocated members than for geographically distributed team members.
Shared knowledge of the team	P4b: Shared knowledge of the team helps team coordination in software development.	P7b: Shared knowledge of the team is more important for geographically distributed team members than for collocated team members.
		Proposition 8: Shared knowledge of the team helps offset some of the negative effects of geographic dispersion on coordination.
Task awareness	P5a: Task awareness helps team coordination in software development.	P9: Presence awareness is more important for coordination with geographically distributed team members than with collocated members.
		P10: Presence awareness can offset some of the negative effects of geographic dispersion on coordination.
Presence awareness	P5b: Presence awareness helps team coordination in software development.	

Conclusions and Limitations

THIS RESEARCH HAS SOME POTENTIAL LIMITATIONS. One is that the study was conducted at a single organization with one large, relatively stable software development team. Some of our findings may not readily generalize to other kinds of organizations or to software teams that have a high turnover during the execution of a project, although we speculate that problems stemming from lack of team knowledge will be exacerbated when turnover is high. In addition, although we selected two sites that had only one hour of time zone difference and fluency in English, cultural differences exist between developers in Germany and the United Kingdom, so it is possible that some of the observed effects were due to cultural differences. However, our on-site experience suggests that the German developers were very skilled in their use of the English language and that most developers at both sites were knowledgeable of each other's cultures and traveled frequently to each other's sites. Nevertheless, our study provided much needed insights into coordination issues relevant to the global software development context, and our propositions can be validated in studies of other organizations. Also, while the high reliability of coding with an independent coder is reassuring, the interpretation of results is always subject to construction by the researcher. This problem was mitigated by the fact that four researchers discussed these interpretations and reached the same conclusions. Finally, our study is limited to a few types of team knowledge, but other ones such as "collective mind," "mutual knowledge," and "environmental awareness" may also have an effect on coordination. Despite these possible limitations, we believe that this research makes significant contributions to both the research literature and the practitioner community, as we discuss below.

Consistent with prior research on global software teams [38], our study found that it is more difficult to coordinate tasks across sites than within a single site. While this result may seem unsurprising, the empirical evidence in the extant literature is inconclusive because of the many confounding variables that correlate with geographic distance. Furthermore, as collaboration technologies aimed at bridging distance continue to improve, the effects of geographic dispersion need continual reevaluation. Our results suggest that geographic distance continues to be a problem for coordination. At the same time, our study revealed that team members require a different mix of team knowledge mechanisms to coordinate their work, depending on whether collaborators are collocated or separated by distance. Furthermore, we found that some of the negative effects of geographic dispersion can be offset with knowledge-based mechanisms such as shared knowledge about team members and presence awareness. We found that all the forms of team knowledge we studied were believed to be important for coordination, and some more than others. Shared knowledge was more important for coordination among our participants than team awareness.

Interestingly, shared knowledge of the team was more important for members working across sites, whereas shared knowledge of the task was more important with members at a single site. It is important to note that we are not discounting the importance of shared task knowledge in collaborations across sites or the importance of shared knowledge of the team in collaborations within a site. On the contrary, we

found all types of shared knowledge important in all contexts. However, the relative importance of each type of shared knowledge shifted depending on whether members were referring to collocated or distributed work. We believe this is due to two reasons. First, participants were more likely to discuss problems that were more salient to them. Therefore, in collocated contexts where team members were more likely to know each other well, shared knowledge of the team is less likely to be a problem, so members were more inclined to discuss problems with shared task knowledge. Similarly, in distributed contexts, limited knowledge of teammates at other sites appeared to be more salient and, therefore, members were less inclined to discuss problems with shared task knowledge. Second, as mentioned earlier, this organization used a configuration management system in which developers can record important task issues as they encounter them. As other studies have shown, developers use configuration management systems not only to manage parallel software changes but also to record task information [35], thus reducing the need for distributed team members to rely on shared task knowledge.

While only one participant in collocated work mentioned presence awareness as important in helping coordination, all participants raised this issue in the context of geographically distributed work. This finding supports the efforts of tool developers, who are increasingly incorporating more presence awareness features into their products. We were somewhat surprised that task awareness appeared to have little effect on coordination, but we attribute this finding to the sophisticated software tools employed in the organization we studied, which provides effective task status information to developers. The fact that a configuration management system helped the team coordinate [27, 35] supports the argument that task awareness facilitates coordination. In sum, our findings imply that large-scale global software development organizations can benefit substantially from promoting the use of practices and tools that strengthen various types of team knowledge.

Finally, our study suggests that different people report different types of coordination problems, depending on the types of dependencies they need to manage to do their jobs. Technical groups are more concerned with technical dependencies, whereas managers are more concerned with process and temporal dependencies. This has important implications for practitioners and researchers. Practitioners need to be aware that different tasks have different types of dependencies, thus requiring different types of coordination. Therefore, the dependencies affecting different groups in large-scale complex tasks such as global software development need to be carefully studied and understood before prescribing coordination remedies. Similarly, researchers of team cognition and coordination need to be aware of the fact that different types of coordination mechanisms or team knowledge have different effects on coordination, and that the presence of one particular mechanism (e.g., a configuration management system) may make some forms of team cognition (e.g., task awareness) less necessary and vice versa. Therefore, it is important to control for the presence of multiple coordination mechanisms and forms of team cognition when studying coordination, particularly in global and highly interdependent contexts.

This study is an important contribution for information systems research because understanding how team knowledge affects coordination in teams can help us identify effective system solutions to knowledge management, which can foster larger knowledge structures, greater knowledge sharing, more effective member interconnection, easier access to knowledge sources in the team, and more coordinated collective action [2, 67]. The present study also represents an important contribution to the empirical body of literature on team knowledge, coordination, and geographically distributed collaboration because most of the prior research on team knowledge has been theoretical, and most of the empirical work has taken place in laboratory experiments using real-time (i.e., synchronous) simulated tasks. Our study is the first one to jointly explore the effect of different types of team knowledge on coordination in an asynchronous context in a global software organization.

This study also contributes to practice. Even though there are many collaboration tools that support distributed and asynchronous collaboration, we know very little about the effectiveness of these tools. Tool developers have sensed the potential that awareness features such as presence awareness can have in geographically distributed collaboration, and are flooding the market with related tools. Our findings in this study helped us identify and better understand key features for the next generation of collaboration tools based on team cognition. Some of these features (e.g., to facilitate team mental model formation) have not been implemented in tools. For example, this study suggests that configuration management systems are effective collaboration tools in the global software development domain, providing distributed software teams with shared task knowledge and task awareness to help them manage simultaneous software changes to different parts of the software product. To help organizations become and remain competitive in such aggressively competitive global markets such as telecommunications, our findings imply that better collaboration tools need to be implemented with a wider set of features to facilitate the use of different types of team cognition that can help distributed teams effectively handle a wider range of dependencies.

Acknowledgments: The authors thank Audris Mockus and F. Javier Lerch for their valuable advice and guidance at various stages of this research project. They also thank Yuqing Ren for doing the external coding of the data. Finally, the authors thank the *JMIS* editor and reviewers for their excellent feedback and suggestions during the review process. An earlier research-in-progress version of this paper was presented at the International Conference on Information Systems (ICIS), 2001, New Orleans.

NOTES

1. Ten managers (six technical managers and four project managers), 22 technical staff (four architects and designers, 13 developers, three testing engineers, and two other software professionals), and four product managers.

2. A configuration management system is a sophisticated software tool that helps organize the work flow; keep track of who is working or has worked on particular software files; make simultaneous changes by different developers to various parts of the software code; manage and control software versions; and annotate comments and observations about the code [35].

REFERENCES

1. Adams, M.; Tenney, Y.; and Pew, R. Situation awareness and the cognitive management of complex systems. *Human Factors, 37,* 1 (1995), 85–104.

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

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

4. Andres, H.P., and Zmud, R.W. A contingency approach to software project coordination. *Journal of Management Information Systems, 18,* 3 (Winter 2001–2), 41–70.

5. Anthes, G.H. Software development goes global. *Computerworld Online* (June 26, 2000) (available at http://www.computerworld.com/softwaretopics/software/story/0,10801,46187,00.html).

6. Bohem, B.R. *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice Hall, 1981.

7. Boyer, D.G.; Handel, M.J.; and Herbsleb, J.D. Virtual community presence awareness. *ACM SIGGROUIP Bulletin, 19,* 3 (1998), 11–14.

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

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

10. Bryant, A. Re-grounding grounded theory. *Journal of Information Technology Theory and Application, 4,* 1 (2002), 25–42.

11. Bullen, C., and Bennett, J. Groupware in practice: An interpretation of work experiences. In R. Baecker (ed.), *Groupware and Computer-Supported Cooperative Work: Assisting Human–Human Collaboration.* San Francisco, CA: Morgan Kaufman, 1993, pp. 69–84.

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

13. Cannon-Bowers, J.A., and Salas, E. Reflections on shared cognition. *Journal of Organizational Behavior, 22,* 2 (2001), 195–202.

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

15. Chen, L., and Gaines, B. A cyber-organism model for awareness in collaborative communities on the Internet. *International Journal of Intelligent Systems, 12,* 1 (1997), 31–56.

16. Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20,* 1 (1960), 37–46.

17. Cooke, N.J.; Salas, E.; Cannon-Bowers, J.A.; and Stout, R.J. Measuring team knowledge. *Human Factors, 42,* 1 (2000), 151–173.

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

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

20. Cummings, J. Work groups, structural diversity, and knowledge sharing in a global organization. *Management Science, 50,* 3 (2004), 352–364.

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

22. Deephouse, C.; Mukhopadhyay, T.; Goldenson, D.R.; and Keller, M.I. Software processes and project performance. *Journal of Management Information Systems, 12,* 3 (Winter 1996–97), 187–205.

23. Dourish, P., and Bellotti, V. Awareness and coordination in shared workspaces. In M. Mantel and R. Baecker (eds.), *ACM Conference on Computer Supported Collaborative Work.* New York: ACM Press, 1992, pp. 107–114.

24. Endsley, M. Toward a theory of situation awareness in dynamic systems. *Human Factors, 37,* 1 (1995), 32–64.

25. Espinosa, J.A.; Cadiz, J.; Rico-Gutierrez, L.; Kraut, R.E.; Scherlis, W.; and Lautenbacher, G. Coming to the wrong decision quickly: Why awareness tools must be matched with appropriate tasks. In T. Turner, G. Szwillus, M. Czerwinski, F. Peterno, and S. Pemberton (eds.),

SIGCHI Conference on Human Factors in Computing Systems. New York: ACM Press, 2000, pp. 392–399.

26. Espinosa, J.A.; Cummings, J.N.; Wilson, J.M.; and Pearce, B.M. Team boundary issues across multiple global firms. *Journal of Management Information Systems, 19,* 4 (Spring 2003), 157–190.

27. Espinosa, J.A.; Slaughter, S.A.; Herbsleb, J.D.; and Kraut, R.E. Coordination mechanisms in globally distributed software development. In D. Moitra (ed.), *First International Conference on Management of Globally Distributed Work.* Bangalore, India: GDW Consortium, 2005, pp. 9–20.

28. Espinosa, J.A.; Lee, G.; and DeLone, W. Global boundaries, task processes and IS project success: A field study. *Information, Technology and People, 19,* 4 (2006), 345–370.

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

30. Fish, R.S.; Kraut, R.E.; Root, R.W.; and Rice, R.E. Video as a technology for informal communication. *Communications of the ACM, 36,* 1 (1993), 48–61.

31. Fussell, S., and Krauss, R. Coordination of knowledge in communication: Effects of speakers' assumptions about what others know. *Journal of Personality and Social Psychology, 62,* 3 (1992), 378–391.

32. Glaser, B.G., and Strauss, A. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Hawthorne, NY: Aldine de Gruyter, 1967.

33. Godefroid, P.; Herbsleb, J.D.; Jagadeesan, L.J.; and Li, D. Ensuring privacy in presence awareness systems: An automated verification approach. In W. Kellog and S. Whittaker (eds.), *2000 ACM Conference on Computer Supported Collaborative Work.* New York: ACM Press, 2000, pp. 59–68.

34. Greenberg, S.; Gutwin, C.; and Cockburn, A. Using distortion-oriented displays to support workspace awareness. Technical Report, Department of Computer Science, University of Calgary, Canada, 1996.

35. Grinter, R.E. Workflow systems: Occasions for success and failure. *Computer Supported Cooperative Work, 9,* 2 (2000), 189–214.

36. Gutwin, C., and Greenberg, S. The importance of awareness for team cognition in distributed collaboration. In E. Salas and S.M. Fiore (eds.), *Team Cognition: Understanding the Factors that Drive Process and Performance.* Washington, DC: American Psychological Association, 2004, pp. 177–201.

37. Handel, M., and Herbsleb, J.D. What is chat doing in the workplace? In E. Churchill, J. McCarthy, C. Neuwirth, and T. Rodden (eds.), *2000 ACM Conference on Computer Supported Cooperative Work.* New York: ACM Press, 2002, pp. 1–10.

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

39. Herbsleb, J.D.; Mockus, A.; Finholt, T.; and Grinter, R.E. Distance, dependencies and delay in a global collaboration. In W. Kellog and S. Whittaker (eds.), *2000 ACM Conference on Computer Supported Collaborative Work.* New York: ACM Press, 2000, pp. 319–328.

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

41. Kiesler, S.; Wholey, D.; and Carley, K. Coordination as linkage: The case of software development teams. In D.H. Harris (ed.), *Organizational Linkages: Understanding the Productivity Paradox.* Washington, DC: National Academy Press, 1994, pp. 214–239.

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

43. King, N. Template analysis. In G. Symon and C. Cassell (eds.), *Qualitative Methods and Analysis in Organizational Research.* Thousand Oaks, CA: Sage, 1998, pp. 118–134.

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

45. Kraiger, K., and Wenzel, L. Conceptual development and empirical evaluation of measures of shared mental models as indicators of team effectiveness. In M. Brannick, E. Salas,

and C. Prince (eds.), *Team Performance Assessment and Measurement.* Mahwah, NJ: Lawrence Erlbaum, 1997, pp. 63–84.

46. Krauss, R., and Fussell, S. Mutual knowledge and communicative effectiveness. In J. Galegher, R.E. Kraut, and C. Egido (eds.), *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work.* Hillsdale, NJ: Lawrence Erlbaum, 1990, pp. 111–146.

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

48. Landis, R.J., and Koch, G.G. The measurement of observer agreement for categorical data. *Biometrics, 33,* 1 (1977), 159–174.

49. Lewis, K. Measuring transactive memory systems in the field: Scale development and validation. *Journal of Applied Psychology, 88,* 4 (2003), 587–604.

50. Liang, D.; Moreland, R.; and Argote, L. Group versus individual training and group performance: The mediating role of transactive memory. *Personality and Social Psychology Bulletin, 21,* 4 (1995), 384–393.

51. Malone, T. Modeling coordination in organizations and markets. *Management Science, 33,* 10 (1987), 1317–1332.

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

53. Mann, C.C. Why software is so bad. *Technology Review, 105,* 5 (2002), 33–38.

54. March, J., and Simon, H.A. *Organizations.* New York: John Wiley and Sons, 1958.

55. Massey, A.P.; Montoya-Weiss, M.M.; and Hung, Y.-T. Because time matters: Temporal coordination in global virtual project teams. *Journal of Management Information Systems, 19,* 4 (Spring 2003), 129–156.

56. Mathieu, J.; Goodwin, G.F.; Heffner, T.S.; Salas, E.; and Cannon-Bowers, J.A. The influence of shared mental models on team process and performance. *Journal of Applied Psychology, 85,* 2 (2000), 273–283.

57. Miles, M.B., and Huberman, A.M. *Qualitative Data Analysis: An Expanded Sourcebook.* Beverly Hills, CA: Sage, 1994.

58. Moore, C. IM tools expand presence. *InfoWorld* (September 27, 2002) (available at www.infoworld.com/articles/pl/xml/02/09/30/020930plentim.html).

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

60. Orlikowski, W. Case tools as organizational change: Investigating incremental and radical changes in systems development. *MIS Quarterly, 19,* 3 (1993), 309–340.

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

62. Pauleen, D.J. An inductively derived model of leader-initiated relationship building with virtual team members. *Journal of Management Information Systems, 20,* 3 (Winter 2004–5), 227–256.

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

64. Rentsch, J.R., and Hall, R.J. Members of great teams think alike: A model of the effectiveness and schema similarity among team members. In M.M. Beyerlein and D.A. Johnson (eds.), *Advances in Interdisciplinary Studies of Work Teams: Theories of Self-Managing Work Teams,* vol. 1. Greenwich, CT: JAI Press, 1994, pp. 223–261.

65. Rentsch, J.R., and Klimoski, R.J. Why do great minds think alike? Antecedents of team member schema agreement. *Journal of Organizational Behavior, 22,* 2 (2001), 107–120.

66. Rouse, W.B., and Morris, N.M. On looking into the black box: Prospects and limits in the search for mental models. *Psychological Bulletin, 100,* 3 (1986), 349–363.

67. Schultze, U., and Leidner, D.E. Studying knowledge management in information systems research: Discourses and theoretical assumptions. *MIS Quarterly, 26,* 3 (2002), 213–242.

68. Silvester, J. Attributional coding. In G. Symon and C. Cassell (eds.), *Qualitative Methods and Analysis in Organizational Research.* Thousand Oaks, CA: Sage, 1998, pp. 73–93.

69. Steinfield, C.; Jang, C.-Y.; and Pfaff, B. Supporting virtual team collaboration: The teamscope system. In S.C. Hayne (ed.), *Group '99: International ACM SIGGROUP Conference on Supporting Group Work.* New York: ACM Press, 1999, pp. 81–90.

70. Strauss, A., and Corbin, J. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory,* 2d ed. Thousand Oaks, CA: Sage, 1998.

71. Tashakkori, A., and Teddlie, C. *Mixed Methodology: Combining Qualitative and Quantitative Approaches.* Thousand Oaks, CA: Sage, 1998.

72. Teasley, S.D.; Covi, L.A.; Krishnan, M.S.; and Olson, J.S. Rapid software development through team collocation. *IEEE Transactions on Software Engineering, 28,* 7 (2002), 671–683.

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

74. Tiwana, A., and McLean, E.R. Expertise integration and creativity in information systems development. *Journal of Management Information Systems, 22,* 1 (Summer 2005), 13–44.

75. Van De Ven, A.H.; Delbecq, L.A.; and Koenig, R.J. Determinants of coordination modes within organizations. *American Sociological Review, 41,* 2 (April 1976), 322–338.

76. Vreede, G.-J., de; Jones, N.; and Mgaya, R.J. Exploring the application and acceptance of group support systems in Africa. *Journal of Management Information Systems, 15,* 3 (Winter 1998–99), 197–234.

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

78. Watson-Manheim, M.B.; Chudoba, K.; and Crowston, K. Discontinuities and continuities: A new way to understand virtual work. *Information, Technology and People, 15,* 3 (2002), 191–209.

79. Weber, R.P. *Basic Content Analysis.* Newbury Park, CA: Sage, 1990.

80. Wegner, D. Transactive memory: A contemporary analysis of the group mind. In B. Mullen and G. Goethals (eds.), *Theories of Group Behavior.* New York: Springer-Verlag, 1986, pp. 185–205.

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

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

83. Wellens, R. Group situation awareness and distributed decision-making: From military to civilian applications. In J. Castellan (ed.), *Individual and Group Decision-Making: Current Issues.* Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. 267–291.

84. Wittenbaum, G.M., and Stasser, G. Management of information in small groups. In J.L. Nye and A.M. Brower (eds.), *What's Social About Social Cognition?* Thousand Oaks, CA: Sage, 1996, pp. 3–27.

Appendix A. Face-to-Face Interview Questionnaire

Background

1. What are your main areas of expertise?
2. How many years of experience do you have in these areas?
3. Which modification requests/features have you worked on recently?
4. What proportion of this work was done across sites (relative to collocated)?
5. What were your roles and responsibilities on these modification requests/features?
6. How much prior working experience did you have with your local peers on this work?
7. How much prior working experience did you have with other-site peers on this work?

Group/Task Process Variables

8. Who in your modification request/feature team did you need to communicate with to do your work (a) locally and (b) at other sites?
9. What types of information/knowledge do you need to exchange with them (a) locally and (b) at other sites?
10. What problems do you encounter when trying to communicate, coordinate, or exchange information with them (a) locally and (b) at other sites?
11. How are these problems addressed, or how could they be addressed effectively (a) locally and (b) at other sites?

Appendix B. Coding Template for Face-to-Face Interviews

Effects of implicit coordination mechanisms on coordination (attributional)

IC	Implicit mechanisms help coordination or lack of mechanisms hurt coordination.
IC.CO	Collocated work.
IC.X	Cross-site work.
IC.CO.SM IC.X.SM	Shared knowledge of the task/shared mental model of the task are important.
	Lack of common grounding (or terminology) in technical language.
	Difficulties conveying technical concepts.
	Common knowledge of key concepts, products, processes, etc.
	Need to internalize concepts (e.g., global teamwork, etc.).
	Need to share knowledge, information, etc., on products, parts, processes, etc.
IC.CO.TM IC.X.TM	Shared knowledge of the team/shared mental model of the team are important.
	Know (don't know) who knows what.
	Know (don't know) people's skills, expertise, etc.
	Know (don't know) who is who.
	Know (don't know) who to call.
	Know (don't know) who has the information when needed.
IC.CO.TA IC.X.TA	Task awareness is important.
	Know (don't know) progress on task.
	Know (don't know) if behind schedule.
	Things discussed on informal discussions, events, etc., not recorded anywhere.
	Don't know of relevant discussions others had.
IC.CO.PA IC.X.PA	Presence awareness is important.
	Cannot find people when I need them.
	Know/don't know when people are around.
	Out of sight/out of mind problem.

Contrast other collocated and cross-site coordination issues

OT	Other coordination issues.
OT.CO	In collocated work.
OT.CO.NO	Not a major problem coordinating collocated work.
OT.CO.PRI	Priority conflicts are a problem (e.g., difficult to figure out urgency of project).
OT.CO.OVR	Work overload is a problem, people are too busy, time pressures.
OT.CO.ATT	Getting people's time or attention is a problem, availability of people.
OT.X	In cross-site work.
OT.X.INT	Little opportunity for interaction, can't just walk to someone's office and talk.
OT.X.RCH	Low richness of communication media.
OT.X.OTH	Other problems: time zones, language, cultural differences, etc.
OT.X.DLY	Delays are more substantial when working across sites.
OT.X.KNW	Prior knowledge of cross-site colleagues, context, etc., helps.

	Frequent visits to the other site helps (at least at the beginning).
OT.X.RED	Need redundant roles, redundant or liaison roles help (e.g., liaison engineers).
	Having a counterpart, colleague, etc., to call in other site helps.

Identifying types of coordination: (1) problems experienced; (2) type of information exchanged

CRD	Instances and types of coordination problems.
CRD.TCH	Technical coordination problems (i.e., "what" is being done, technical dependencies not well managed).
	Need to exchange information/knowledge on technical details, specs, etc.
	Problems integrating software/hardware parts, modules, products, etc.
	Software repairs, errors, bugs, etc.
	Inadequate implementation of interfaces among parts, etc.
	Inconsistent designs, changing standards (e.g., ETSI).
	Reliability problems with software produced.
	Incomplete releases of software product.
CRD.TMP	Temporal coordination problems (i.e., "when" it is done, temporal dependencies not well managed).
	Need to exchange information/knowledge on schedules, etc.
	Need to synchronize plans, activities, work, etc.
	Plans are too aggressive for tightly coupled activities.
	Missed delivery dates.
	Unsynchronized (gate) review dates.
	Late work from prior software phases (e.g., design, coding, testing).
CRD.PRC	Software process coordination problems (i.e., "how" it is done, software process dependencies not well managed).
	Need to exchange information/knowledge about processes, how things are/need to be done.
	Duplication or redundant work.
	Software phase started before it was supposed to.
	Not following the established software process.
	Not following what was agreed upon at meetings, discussions, etc.
	Confusion with new software tools (e.g., ClearCase).
	Working in "crisis" mode (i.e., stop working on one thing to address problems).
	Priority confusions and priority conflicts.
	Unresolved blocking issues, need to escalate issues to higher levels.
	Project scope changes, frequent changes in feature list for a software release (i.e., "feature churn").
	Frequent changes in specifications, standards, etc. (e.g., ETSI).

Appendix C. Coding Example

Case: T1-TST-GE (Technical Staff 1, Testing Engineer, Germany)

Input Variables (Team, Task, and Environment)

1. Functional areas of responsibility: testing.
2. Project areas of responsibility: testing, functional testing (Rel 3), feature testing (Rel 4). Then deliver to network testing and integration (NTI) group.
3. Areas of expertise: testing, software development background, telecom, GSM, systems architecture. Software development for 2 megabit interface.
4. Years of experience in these areas: 10 years.
5. Supervise MRs (i.e., modification requests)? Not really. We issue MRs and decide which MRs can be included in the next load based on risks found during testing.
6. Supervise feature development? No.

Group/Task Process Variables

7. Who in your team or workgroup do you need to communicate with to do your work (below, above, and laterally to you)?
   a. Locally: My own team (that I supervise), project management, development team leader, sometimes with developers too, NTI people (my customers), systems architecture team, release manager too.
   b. In other sites: Release test manager, testing team leader in England (my counterpart), quality control people in England, network testing team in India.
8. What types of information/knowledge do you need to exchange with them?
   a. Locally: Information on loads, when are releases planned. Mostly about specifications (feature requirement definition, software decomposition), technical details, very specific to understand what's behind each feature. Process time lines, such as when the next load is planned, and the content of the load. Testing is at the tail end of the process, so if anyone gets behind, I get behind. It happens all the time. Need to know what and when. Similarly, my work feeds into NTI via project management. Need to raise my hand to let them know if I am behind.
   b. At other sites: Information about MRs raised as a result of problems found so that they can start fixing them. Also, about what tests we plan to do and which ones we have done. Interaction is more intense during planning stage to decide what will be done where, and also when I need to use resources from other sites, talking directly to the people I need to work with (haven't used Indian resources yet for my own responsibility in testing).
9. What problems do you encounter when trying to communicate, coordinate, or exchange information with them?

a. Locally: Being last in the process comes with the territory. Need to keep track of what is happening up front so that I am prepared for when issues come up. Getting information about when testing needs to be done is clear, no problem there. There are more problems in the realization of schedules. If things aren't ready on schedule, I need to replan all my team's work. Often schedules are on average one or two days late. This issue has to do with the organization. This happens because software development deadlines are planned too tight, so the software build, which is done by many people, always finds last-minute problems that delay the loading, and therefore the testing. In terms of getting the what (what is in the feature), the problem is mostly on lateness. When this happens, we cannot be prepared for the testing. Locally, if the content is not clear, we can ask around. With some people, the background is so different that it takes a while to begin to understand each other, which sometimes causes miscommunication. It is not so much a problem for me but it is a problem for people with less experience. Not a problem with availability of people.
b. At other sites: Availability of people is more of a problem here. Sometimes the voice mail message is not updated to reflect they are on vacation, so don't know when they will get back to me. When this happens, I call people who sit next to these people to inquire (surrounding neighbors). It works, but this can be improved. It is less of a problem with auto-replies in e-mail. Normally, information about MRs raised and what tests have been done or will be done are not a problem. This information is normally published via Web. Some Webs are better updated than others. We don't have a tool for test management system or for automatic scheduling of test events (or shared databases, project management) to help us on this.

10. How are these problems addressed, or how could they be addressed effectively?
    a. Locally: We simply increase the number of status meetings so that everyone is aware of everyone's needs. I think we now understand our problems better. Therefore, we are able to build contingencies and more realistic schedules. If we could assign more resources to deal with process planning and tracking (or by project management), then we could communicate and define loads more effectively. The delays are usually because we have to do more than we thought. We need more resources from project management to keep things coordinated. Project management should not only be looking for coordination problems but also helping find solutions. We don't need them to tell us that we are late, but we need people that can take workload from us, especially project management activities. We use MS Project software to manage this. Need more action and intervention when project gets off schedule.
    b. At other sites: When problems are found, we make phone calls or we travel. More consistency across Web sites would help.