SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. 2004; **8**: 000–000 (DOI: 10.1002/spip.185)

The Impact of Time Separation on Coordination in Global Software Teams: a Conceptual Foundation

Research Section

J. Alberto Espinosa*[†] and Erran Carmel
Kogod School of Business, American University, 4400 Massachusetts Avenue, N.W. Washington, D.C. 20016-8044, USA

While there has been much research on the study of global virtual teams and global software teams, there has been practically no research on the nuances of time separation. We present three converging perspectives on this topic: (a) a view from practices and tactics of global teams; (b) a theoretical view from coordination theories; and (c) a view from our prior research in which we modeled coordination costs for time-separated dyads. Practice suggests that time separation arises not only from time-zone differences but also from factors such as nonoverlapping weekend days and holidays, shifts, and different working schedules. It also suggests that teams employ various coping tactics when faced with time separation – synchronous, asynchronous, and education. Theory suggests that communication is necessary to coordinate and that effectiveness of communication is hampered, both in quality and timeliness, when teams are separated by time. Our model, based on coordination theory, suggests that coordination costs contain four main components – communication, clarification, delay, and rework – and that the various aspects of time-separated work have different effects on each of these components. Our convergent view from these three perspectives shows that distance separation is symmetric – i.e. distance (A,B) = distance (B,A) – while time separation is asymmetric, which affects the planning of team interactions; that the timing of activities matters in time-separated contexts but not in contexts with only distance separation; and that *vulnerability costs* (i.e. resolving misunderstandings and rework) increase with time separation. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: global software teams; global software development; geographically dispersed teams; coordination costs; time separation

1. INTRODUCTION

Coordination in different-time contexts (time zones, holiday differences) is difficult because of lean communication media, difficulties in resolving unclear messages, reduced opportunities for spontaneous interaction, and lack of contextual reference. Fundamentally, time differences tend to increase *coordination costs*. Yet, despite these costs, team work is increasingly carried out globally. There are a number of reasons for this increase. One reason is that since software products are digital, their transportation costs are very low and delivery time is effectively zero. Also, *production*

* Correspondence to: J. Alberto Espinosa, Kogod School of Business, American University, 4400 Massachusetts Avenue, N.W.Washington, D.C. 20016-8044, USA
[†]E-mail: alberto@american.edu

costs in many ('offshore') distant locations are low. In addition, geographic dispersion enables companies to access specialized software talent and technical resources (Carmel 1999).

These cost-benefit trade-offs – of higher *coordination costs* and lower *production costs* – are important, complex, and not fully understood. As a result, this topic has interested researchers and practitioners studying coordination in distributed software teams (Carmel 1999, Herbsleb and Grinter 1999, Herbsleb *et al*. 2001, Espinosa *et al*. 2002) and geographically dispersed teams in general (Van den Bulte and Moenaert 1998, Olson and Olson 2000, Cramton 2001, McDonough *et al*. 2001, Armstrong and Cole 2002, Kiesler and Cummings 2002). Research focused on time differences has only begun to appear recently (Klein and Kleinhanns 2003, van Fenema and Qureshi 2004).

There are a number of difficulties associated with the study of global software teams, particularly when trying to understand the effect of geographic dispersion. For example, many studies look at geographic dispersion as a binary attribute – i.e. teams are either colocated or geographically distributed. However, teams may operate in a variety of geographic dispersion configurations (O'Leary 2001, O'Leary and Cummings 2002) (e.g. two sites: one central site with several small satellite sites, several sites with evenly distributed effort, etc.).

On the basis of the configuration permutations of O'Leary and Cummings (O'Leary and Cummings 2002), we discuss three cases of increasingly complicated time adjustments, illustrated in Figure 1. First, two sites working in different time zones separated by a few hours (e.g. England–Germany, New York–Chicago) can mutually adjust their work schedules such that they maximize work-time overlap. Second, one hub site (e.g. London) with many developers collaborating with a number of developers in multiple satellite locations spread throughout multiple time zones (e.g. New York and Bangkok). Thus, developers in the satellite locations can adjust their work hours to maximize overlapping work hours with the central hub location. Third, and most difficult, is when many developers are widely scattered across multiple time zones, providing very little work-time overlap in which developers can interact simultaneously.

Researchers have found that difficulties due to geographic dispersion often correlate with other team boundaries like functional identity, differences in local context and local culture, etc. (Orlikowski 2002, Watson-Manheim *et al*. 2002, Espinosa *et al*. •2003). More specifically, we emphasize that when distributed teams are also separated by time (e.g. time zones, differences in work cycles, shift work, etc.) it becomes difficult to tease out the true effects of geographic dispersion. Distance and time effects are often confounded in global software team studies because many geographically dispersed teams are often also separated by time zones. AQ2

In this paper, we discuss important conceptual issues and analyze the implications of time separation from three perspectives. We first discuss time-separation issues from a practical perspective. We then discuss similar issues from a theoretical perspective. Because of the paucity of research on the effects of time separation, we bring to bear theories related to coordination in general and the research literature on coordination in software development. We then analyze the implications of time separation from these theoretical perspectives. Finally, we present our coordination model to better understand the effects of time separation on coordination in software tasks. We conclude with a discussion section where we identify overarching issues derived from these three perspectives, which affect research and practice in time-separated contexts, and then offer suggestions for further research.

2. TIME-SEPARATION ISSUES: A VIEW FROM PRACTICE

In this section, we summarize tactics (in Table 1) that we have found from interviews conducted for other studies of global software teams (Carmel 1999, Espinosa 2002) and through exploratory interviews and discussions we have conducted recently with software professionals involved in time-separated collaborations. The interviews were taped and transcribed in each of the studies. We analyzed the data by identifying incidents in which interviewees brought up time-separation issues. Our method is consistent, to some extent, with the Critical Incidents method (Chell 1998), but we departed from it in some respects: the interview questionnaires were semistructured and were designed for other studies; formal interview

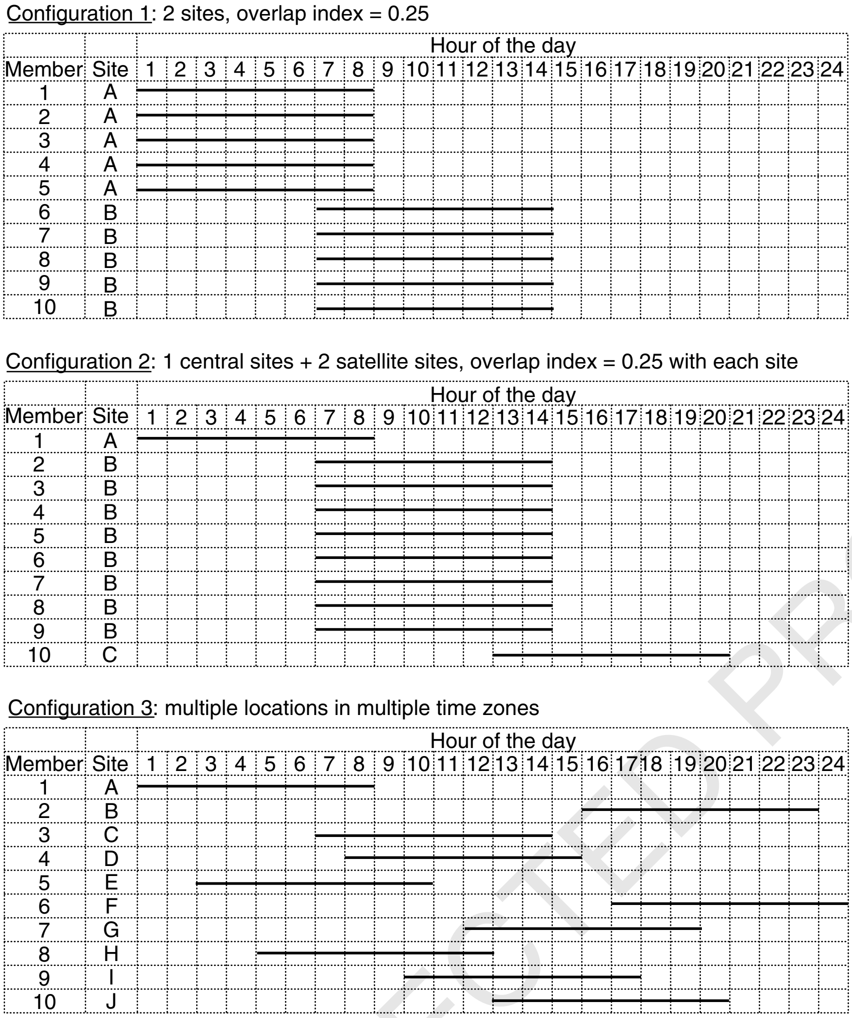

Figure 1. Different time-separation configurations

data was complemented with data from other more informal interviews; and, because our objective was to explore issues and present a conceptual framework to study time separation, we made interpretations of events described involving time separation. Because the previous studies that we used as a reference for this study were about global software teams, incidents involving geographic dispersion and time separation were abundant.

2.1. Practices Used by Virtual Teams to Overcome Time Separation

In overcoming time-zone differences, we found three principal solution tactics, which we summarize in Table 1 and discuss in more detail below:

- *Asynchronous*: Teams instill better practices in their nonoverlapping work times to compensate for the lack of common work hours.
- *Synchronous*: Teams plan for the existing synchronous overlap times and/or enlarge the windows of synchronous (overlapping) times.
- *Education*: Individuals in the teams become more effective across time differences, with better awareness and better information.

Teams use a number of *asynchronous* tactics to cope with time separation. First, and most obvious, they make better use of asynchronous technologies, such as electronic mail, voice mail, and use of various shared databases and other repositories (groupware, knowledge management, team intranets and

Table 1. Tactics to overcome time separation

Category	Tactic
Asynchronous	• Use of asynchronous technologies • Formalization of activities • Bunch-and-batch • Break the e-mail chain
Synchronous	• Shift dialogue to overlap time and independent work to asynchronous time • Expand overlap window by working longer • Expand overlap window by shifting work hours • Expand overlap window by always being available • Create liaison roles with adjusted hours that expand overlap window of only• those individuals rather than the entire team
Education	• Build awareness • Create easy access to current time, calendar, and holiday schedule of distant actor(s)

web sites, discussion areas, etc.). Domains like software development also have specialized collaboration tools designed to help team members work effectively in an asynchronous way (e.g. configuration management systems, error logs). For example, in one of our previous studies, we found that a substantial amount of coordination in distributed software teams was accomplished through a configuration management system (Espinosa *et al*. 2002). Such systems are generally used to help developers manage simultaneous software changes, but many developers in that study used the comments field to exchange asynchronous notes and messages about the code. This has also been observed in other studies (Grinter 2000). Effective time-separated teams also learn to formalize (i.e. program, structure) activities and messages so that they convey information in a more effective manner, thus reducing the need for further clarification communication. They also learn to organize their workdays so that they bunch-and-batch their work in order to maximize completion, before the work is delivered to distant sites. Finally, effective individuals learn to 'break the e-mail chain'. The e-mail chain begins when one actor initiates a message, the receiver does not understand it fully and asks for clarification, the sender attempts to clarify, the receiver misinterprets again, and so on. Meanwhile, an entire week has gone by. Therefore, experienced individuals stop this chain early 'by picking up the phone' and clarifying the message through a richer communication medium.

Synchronous tactics address time separation more directly. First, if there is some time overlap, teams synchronize their dialogue time so as to maximize synchronous exchange (e.g. telephone, instant messaging, videoconferencing). Thus, work that can be done independently is conducted during nonoverlap time so that overlap time can be devoted to meetings, telephone conversations, adjustments, problem resolutions, and other actions better done synchronously. There are a number of variations on this tactic noted in the field data of Klein and Kleinhanns (2003): experienced actors learn to package their information so that it can be better absorbed by the distant actors, more work is shifted to nonoverlap time so that synchronous meetings become more productive, and questions for overlap time are prepared ahead of time. Second, and most familiar, teams tend to enlarge the overlap period by shifting and expanding work hours. For example, European staff may start late and work late so as to have greater overlap with their American counterparts. Conversely, the American staff may start early so as to expand the overlap time with their European counterparts. Japanese companies are notorious for working late hours, thus enlarging the overlap window with their counterparts. We heard recently from a Chinese software company developing software for a Japanese client (China is one hour behind Japan) that because Japanese developers tend to work late, there is no noticeable time difference.

Some software organizations also create liaison roles to help team members interact across sites. In one of our previous studies involving a software team with members in the United Kingdom, Germany and India, we found that a number of Indian software engineers were trained in the UK and German sites for a few months to familiarize themselves with team members and the work context in those sites and then worked as liaison engineers (Espinosa 2002). Once trained, these liaison engineers would go back to India and would serve as points of contact for the UK and German developers. Liaison engineers would often adjust their work schedules to increase their window of work-time overlap with their British and German counterparts. In practice, this time-window expansion is practiced only by some of the virtual team members – particularly managers and team leaders.

Time *education* tactics involve learning how to work effectively under time-separated conditions. Less-experienced team members need to be made aware of time-separation issues. They are not used to thinking about their counterparts being gone for the day while they work. They are not used to computing the direction of the time difference. Thus, various awareness tactics are important. (e.g. the distant team member reminds her counterpart that the scheduled meeting is set for 2 PM local time, and members remind their distant teammates about shift to 'daylight savings time', which is at different times in different countries). A simple tactic is to post hours and time differences on the common web site.

2.2. Time Separation is Disruptive

One often hears that individuals in global software teams spend many evenings, nights, and early mornings in telephone conversations across the oceans. Overlap-window expansion, which we noted above, is a disruption of one's personal time and further dilutes the boundaries between work and home life. Now that wireless communication devices are ubiquitous, key individuals are always reachable. Balanced teams try to shift the burden of late-night (or early-morning) conference calls in order to soften the pain of disruption. But, we have heard of many cases where the dominant/hub site dictates meeting times convenient to their normal workday, never adjusting for the sake of the distant participants. We note a similar litany of complaints about time differences in the work of Klein and Kleinhanns (2003) and van Fenema and Qureshi (2004). Not all individuals are accommodating on overlap windows. We heard a story at one major California-based technology firm working on urgent software fixes in a global collaboration, in which

> 'the British technical experts liked waking up early in the day to work, while their California counterparts liked coming into the office late and working late (California is 8 hours behind Britain). Thus, they had no synchronous overlap window and relied on one e-mail batch per day, which really slowed down the work.'

2.3. Time Separation is more than just Time-zone Differences

While time-zone differences are the most recognizable element of time separation in work coordination, other factors increase coordination difficulty: work hours, lunch breaks, weekend times, and holiday times. These are summarized in Table 2. We discuss each of these in turn.

Time-zone differences, even small ones, can create substantial problems if the work-time overlap between the two sites is not synchronized. For example, a study on coordination in global software teams found that a one-hour time-zone difference between two sites substantially affected the team's ability to communicate interactively because it reduced their overlapping time by four hours – one hour at the beginning of the day, one hour at the end of the day, and one hour during each site's lunch break (Grinter *et al.* 1999). Work hours may also vary by country. While Americans are used to a standard day of roughly nine-to-five, office workers in Spain start working later in the day, have longer lunch breaks, and finish their workday often much later than 7 p.m.

Weekend times may also vary. While much of the world has a weekend on Saturday and Sunday, this is not universal. In Arab countries, Friday is not a workday. The weekend in Israel is Friday and Saturday, which creates a long 'blackout period' when working with collaborators in the United States – Americans come to work on Thursday morning when the Israelis have already left for the weekend. The patchwork of national holidays is also bewildering. One American technology firm we interviewed had staff in more than a dozen European nations and because of different national holidays, there were only 50 regular workdays in

Table 2. Types of time differences

- Time-zone differences
- Workday differences (i.e. start and ending times of workday)
- Weekend differences (i.e. weekend days vary)
- Holiday differences (i.e. religious and national holidays)
- Lunch and other break hours (e.g. Americans break for lunch earlier than many other cultures).

common in any given year for the purpose of scheduling synchronous meetings (e.g. the entire month of August is not usable for several European nations).

2.4. Configuring Global Software Teams for Time Separation

We note two global team configurations that specifically address time separation. We emphasize that, unlike the practices described earlier, they are not tactical in nature. The first purposefully positions teams in nonoverlapping time zones, while the second purposefully positions teams in overlapping time zones. The first is an approach that has received a great deal of attention: *'Follow-the-sun'* work, also known as *'round-the-clock'* software development (Carmel 1999), which takes advantage of time-zone differences to speed up project work. For example, a team in Eastern United States can hand off work at the end of their day to team members in India or China, who can continue the task after the US team members go to sleep. The appeal of this strategy is enormous, for, if it can be coordinated properly, it can reduce project duration by a factor of two for the two sites mentioned above, at least in theory. Clearly, coordination in *follow-the-sun* must be effective, which is why the authors are not aware of any successful cases of this approach on a regular basis. Many teams have noted occasional time reduction using the *follow-the-sun* approach (e.g. once a week). But, continuous *follow-the-sun* is too difficult for software teams to conduct because of the high dependencies implicit in the concept and the need for near-perfect communication and coordination. However, we have found *follow-the-sun* to be effective for low granularity tasks such as bug-fixing or call-center activity (e.g. technical support).

The second configuration approach is the purposeful positioning of a companion site within closely overlapping time zones. Gumpert (2004) describes a case of a software start-up in Austin, Texas that started collaborating with an offshore partner in India (whose time zone is 11.5 hours ahead). The principals at the firm found that coordination with India was too difficult because of time differences, and they moved to an offshore partnership in Columbia – only one time zone away from Texas.

3. COORDINATION IN SOFTWARE DEVELOPMENT: A VIEW FROM A THEORETICAL PERSPECTIVE

As we noted, there has been no theoretical research specifically on the impact of time differences. AQ5 Thus, we •turn to coordination theories to inform the deeper understanding of the impact of time differences. Consistent with coordination theory research, we define coordination as the management of dependencies among task activities to achieve a goal (Malone and Crowston 1990, 1994). A few important principles deriving from this definition are worth noting. First, if task activities can be carried out independently, then there is no need to coordinate. Conversely, more complex tasks like software development have substantial dependencies that need to be managed, thus the need for coordination. For example, when many software individuals and teams are working in parallel to build a single software product, different software parts need to interoperate properly and tasks (e.g. coding) need to be completed on schedule to avoid delaying other tasks (e.g. testing). Second, when task activities contain tightly coupled dependencies, the individual decisions and actions of team members involved in a task become mutually constraining (Herbsleb and Mockus 2003). One team member's work on a task may need to stop until another team member's work is completed. Finally, if a task is analyzed with a fine-grained level of detail such that the dependencies and mutual constraints among task activities are well understood, one can begin to identify different coordination mechanisms that can be employed to manage these dependencies effectively.

Dependencies in a task can be pooled (i.e. two tasks depend on the same resource pool), sequential (i.e. task A cannot proceed until task B is completed), or reciprocal (i.e. tasks A and B are interdependent) (Thompson 1967). For example, one team member may be working on a task (e.g. software coding) and may reach a point at which the work needs to be handed over to another team member who needs to perform another task (e.g. testing) such that the first member's work on this task cannot continue until the second member's task is finished. This sequential dependency among two members needs to be effectively managed to achieve coordination.

The organizational research literature suggests that team members coordinate nonroutine aspects of their work through communication (March and Simon 1958, Thompson 1967, VanDeVen *et al*. 1976). When team members are separated by geographic distance and/or time, their ability to communicate interactively and on a timely basis is hampered, thus negatively affecting team members' ability to manage dependencies among their task activities. Thus, while teams also use other coordination mechanisms (e.g. plans, tools), we focus our discussion on coordination by communication because software development is a complex task with substantial nonroutine, interdependent activities, which require a fair amount of communication to coordinate. In addition, communication is an obvious way for team members to generate other coordination processes (Malone and Simon 1994). Furthermore, communication is important in time-separated contexts because the frequency (Allen 1977, Kiesler and Cummings 2002) and (Waller 1999, Gittell 2001) timeliness of communication can be adversely affected when team members are not in close proximity.

On the other hand, a recent study found evidence that software teams working in the same room had significantly higher productivity than other teams that were not colocated (Teasley *et al*. 2002). They concluded that their productivity was greater because colocation in the same room bolstered collaboration by facilitating interactive continuous communication and awareness. In contrast, one may conclude that when team members are separated by distance, these benefits disappear. Furthermore, if they are also separated by time differences, then both continuous communication and awareness of team members will be hampered even more, thus causing further delays because of coordination breakdowns and rework, making it particularly difficult to close open issues. As another study found, spanning multiple time zones can affect the rhythm of a team's work, creating unexpected faultiness (Espinosa *et al*. 2003), more so if teams are separated by additional boundaries (e.g. culture, function, language) (Lau and Murnighan 1998).

4. OUR COORDINATION MODEL: A VIEW FROM A MATHEMATICAL PERSPECTIVE

In this section, we describe our model of *coordination costs* due to time differences in dispersed software teams. Our model is more fully described and validated with simulated data elsewhere (Espinosa and Carmel 2004). While our focus is on global software teams, the model is generic, and can apply to any type of virtual knowledge team. Our model is derived following Malone and Crowston's coordination theory, (Malone and Crowston 1990, Malone and Crowston 1994) in which coordination is viewed as the management of dependencies among task activities, and Malone's formulation of coordination costs in organizations and markets (Malone 1987). While coordination theory does not specifically address issues of distance and time separation, we incorporate distance and time separation in our analysis by evaluating how the total cost of carrying out a task is influenced by the cost and effectiveness of different communication mechanisms in various collaboration modes (i.e. colocated and separated by distance and/or time) and by delays caused by time separation.

We begin by delineating our assumptions about distributed coordination and communication. First, we make no distinctions between the granularities of a task request encapsulated in a message. A task can be a large one, perhaps requiring several days of effort, or it can be a very small one, such as a yes–no answer. Next, we make an assumption about media choice. If a situation arises in one site that requires interaction with another site during their off-work hours, being unable to pick up the phone and call other members can slow down a group's progress. The choices for a team member in such a situation are to either send a request asynchronously (e.g. e-mail) or wait until work hours overlap again to make the request synchronously (e.g. phone call). Requests are often not clear, requiring additional clarification communication, further delaying the whole process. When team members are working face-to-face, the clarification may be nearly instantaneous. Even when members are distant, but in same-time zones, clarifications can be made very quickly through phone calls, •IM, or videoconference. AQ6 However, when team members are separated by time, the need to clarify messages will introduce further delay, unless this happens during work-overlapping hours.

Our model begins by looking at a single collaboration act between two actors – a task *Requestor* who makes a request to another actor who is the task *Producer* because of a workflow dependency, AQ7 i.e. the work of the *Requestor* cannot continue until the work of the *Producer* is finished. •For this to happen, the *Requestor* must communicate the task requirements to the *Producer*, and the *Producer* must communicate an acknowledgement to the *Requestor* when the dependent task is completed (Malone and Crowston 1994, Malone *et al*. 1999). As illustrated in Figure 2, team members may be interacting in any of four possible (2 × 2) collaboration modes, depending on whether the dyad is separated by distance and/or by time: face-to-face, separated by distance only, separated by time only, or separated by distance and time (Bullen and Bennett 1993).

The coordination issues of two such actors, who are separated by distance, can be substantial. Our model shows that these issues compound even further with time separation. For example, one central aspect of our model is that the overlap in work hours between any two members who collaborate can take place either at the beginning or at the end of one's workday. The synchronous or asynchronous solutions to time separation will have to be worked out differently, depending on when the work-time overlap occurs in one's workday.

In our model, actors need to communicate, and this communication is costly and time separation introduces asymmetries. An asymmetry takes place when work overlap occurs at the beginning of one site's workday and at the end of the other site's workday (there is no asymmetry when work times fully overlap). Because of this asymmetric property of time separation, we argue that the effect of time separation on global software team coordination can be modeled and studied by analyzing timing issues (e.g. when interactions occur, task duration times, and amount of overlap in work hours) and then by evaluating how they affect *production costs* (i.e. the cost of carrying out individual tasks) and *coordination costs* (i.e. the cost of managing the dependencies between individual tasks).

This breakdown of total costs into production and *coordination costs* is similar to the breakdown suggested by Malone in his theoretical modeling of coordination in organizations and markets (Malone 1987), which has been widely used in theoretical and simulation research involving coordination (Koushik and Mookerjee 1995, Carley and Lin 1997, Jehiel 1999). However, for the purposes of studying the effects of time separation, we find that it is more useful to further decompose *coordination costs* into: (a) *communication costs* – the cost of maintaining communication links and the cost of sending and receiving messages; (b) *delay costs* – the cost of delays caused by the dependency requiring communication; (c) *clarification costs* – the cost of further communication required to repair miscommunication; and (d) *rework costs* – the cost of further production necessary for work that was completed before the miscommunication was discovered (see Table 3). Following Malone's terminology, we refer to clarification plus *rework costs* as *vulnerability costs* because these costs originated as a result of miscommunication. *Delay costs*, on the other hand, are affected by the latency inherent in the communication media and by working-time differences.

While our model follows Malone's model, we make some adjustments to take into account delays resulting from distance separation or time zones differences. First, we specifically model time and

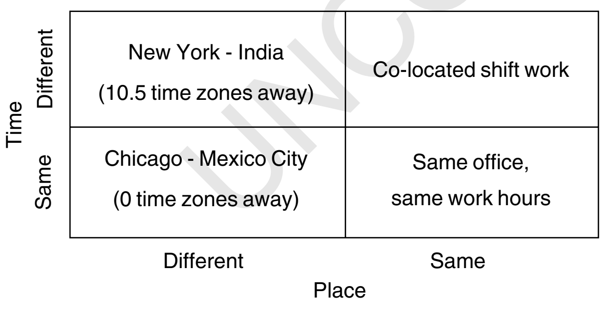

Figure 2. Time by place matrix

Table 3. Cost components

Cost components	Definition
Production	The costs of carrying out the task
	Coordination costs
Communication	The costs of maintaining communication links and sending and receiving messages.
Delay	The costs incurred because one actor is waiting for another to begin their work day.
Clarification	The additional cost of communication and delay due to prior miscommunication that resulted from the need to communicate asynchronously.
Rework	The additional costs of production due to miscommunication that resulted from the need to communicate asynchronously.

distance separation between actors. Second, Malone's model analyzes different coordination structures for a set of actors, while our model employs only two actors, who need to carry out a task with a tightly coupled workflow dependency, who coordinate via communication. Finally, Malone's model assumes that actors employ their production capacities optimally, but we do not need to make this assumption because there are only two actors in our model.

Malone defines *production costs* as the average delay in processing the task, but since Malone's model does not incorporate time delays due to time separation, his *production costs* amount to the time it takes to carry out the task, which is consistent with our definition of *production costs*. Malone defines *coordination costs* on the basis of the cost of maintaining communication links and the cost of sending messages among nodes in the coordination structure. However, in Malone's model, messages arrive instantly. Our definition of *coordination costs* is similar to Malone's but we also incorporate the time delay introduced due to time separation (e.g. one member may send a task request during the other member's off-work hours). Finally, Malone defines *vulnerability costs* as those due to failures of those involved in the task, leading to task reassignments. Because our model involves only one dyad, there is no reassignment. Instead, failures lead to further communication and coordination to clarify things and, possibly, to reprocess part of the task (i.e. rework). A message can be unclear, with some probability. Unclear messages can lead to either: (1) rework, resulting in additional *production costs* for a portion of the work with further delays; and/or (2) a simple request for clarification, resulting in additional *coordination costs*. We now describe the mathematical formulation of the main components of our model. All cost variables are specified in financial terms and all time variables are specified as proportions of a workday (e.g. 0.5 = half of a workday, 3 = two workdays).

Production Costs (Pc) in our model are simply the *Producer's* daily cost of carrying out tasks, and it can be specified as

$$Pc = \lambda Cp Tt \quad (1)$$

Where λ is the daily frequency of task arrivals, Cp is the daily production cost rate for the *Producer*, and Tt is the time it takes the *Producer* to complete the task. This cost component only involves individual production time and costs incurred by the *Producer* and it is unaffected by time or distance separation.

Communication Costs (Cc) for two actors include the daily cost of maintaining a communication link (Cl), plus the daily cost of sending individual single messages (Cm). The cost of maintaining a face-to-face link and the cost of face-to-face communication are assumed to be negligible for colocated teams, compared to other *communication costs*. The cost of maintaining a synchronous and an asynchronous communication link are (Cls) and (Cla), and the cost of sending a synchronous and an asynchronous message are (Cms) and (Cma), respectively. Thus, the daily *communication costs* can be specified as

$$Cc = Cl + 2\lambda Cm \quad (2)$$

that is, a task requires a message to request the task and a message to acknowledge completion of the task. Depending on whether the *Requestor* and *Producer* communicate synchronously or asynchronously, there are several permutations of Equation (2). For example, if both members communicate synchronously, the *communication costs* would be $Cl + 2\lambda Cms$, but if one communicates synchronously and the other asynchronously, the cost would be $Cl + Cla + \lambda(Cms + Cma)$.

Delay Costs (Dc) in our model are measured from the perspective of the task *Requestor*, because this is the actor who has a dependency, whose work is delayed while the *Producer* completes the task. Thus, daily *delay costs* can be specified as

$$Dc = \lambda Td Cd \quad (3)$$

Td is the delay experienced by the task *Requestor* while the *Producer* completes the task and Cd is the daily rate of cost delay for the task *Requestor*. One interesting property of this cost component is that if the *Producer* carries out the task during the *Requestor's* off hours, *Td* is zero, which is the motivator for software work organized in *follow-the-sun* arrangements. On the other hand, if the *Producer* does all the work during overlapping work hours, *Td* is identical to the time it takes to carry out the task *Tt*. Thus, the degree of time separation or work-time overlap for a dyad will have a substantial impact on *delay costs*.

Clarification Costs (*Cf*) will be incurred when task-request messages are not clear and the task *Requestor* and task *Producer* need to communicate again to resolve the misunderstanding, thus incurring further communication and delay costs. If there is a probability *Pu* that a task-request message will be unclear, then *Cf* can be specified as

$$Cf = Pu(Cc + Dc) \tag{4}$$

Rework Costs (Rc) will be incurred if the need for clarification occurs after the *Producer* has started to work on the task and some of the software work needs to be redone. If there is a probability *Pr* that a given unclear message will lead to rework and that the proportion of the total task that needs rework is *Rw*, then *Rc* can be specified as:

$$Rc = PuPrRwPc \tag{5}$$

Clarification and *rework costs* are equivalent to what Malone calls 'vulnerability' costs. In other words, if all goes well, the cost incurred in carrying out the task equals *Pc*, *Cc*, and *Dc*. If these were the only costs incurred, then *follow-the-sun* and *round-the-clock* programming arrangements would be ideal because they would save substantial *delay costs* by maximizing the amount of task production that takes place during the *Requestor's* off hours. However, the problem with these work arrangements surfaces when vulnerabilities materialize, requiring further communication to clarify issues and possible rework. An important aspect of these two cost components is that they are both affected by the quality and richness of the communication medium used to communicate. In our model, the value of *Pu* is dependent on the particular medium used. For example, *Pu* for face-to-face communication is very low because team members have a very rich communication medium that allows them to use contextual references and nonverbal cues. *Pu* is likely to increase as teams move to leaner communication media like videoconference, voiceconference and electronic mail. *Pu* will also increase as global team members span more boundaries (e.g. cultural, functional, language), making it more difficult for members to communicate clearly (Watson-Manheim *et al*. 2002, Espinosa *et al*. 2003).

If, for example, a distributed team communicates via inexpensive voiceconference, then the lean communication media will make it more difficult to convey ideas clearly. On the other hand, if the team uses videoconference with supporting tools (e.g. a whiteboard), the need to clarify messages will be reduced. These two cost components will also be affected by time separation because this may introduce longer delays and may force a team to use asynchronous communication tools at times when such communication media may not be the most effective for the task at hand. As Media Richness theory suggests, lean communication media (e.g. electronic mail) may not be the most appropriate form of communication for equivocal tasks that contain more uncertainties (Dennis and Kinney 1998). We argue that it is these *vulnerability costs* stemming from clarification and *rework costs* that make work arrangements like *follow-the-sun* so difficult for many software tasks.

In sum, our model is parsimonious and it involves individual *production costs* (*Pc*) necessary to carry out individual software development task activities and a coordination cost (*Co*) necessary to manage the dependencies among different task activities. These *coordination costs*, in turn, are composed of *communication costs* (*Cc*), *delay costs* (*Dc*), *clarification costs* (*Cf*), and *rework costs* (*Rc*). Nevertheless, the specific application of these formulas will vary substantially in complexity depending on the pattern and timing of team members' synchronous and asynchronous interaction, as illustrated in Figure 3. One of the key issues that our model uncovers, as depicted in this figure, is that *coordination costs* are sensitive to the time at which a request is initiated and the time at which that request is responded to. A request can be initiated during overlap and be responded to after overlap, it can be launched before overlap and responded to after overlap, or it can be initiated and responded to within the overlap.

Also, while our simple model considers only two actors, a task *Requestor* and a task *Producer*, it can be readily extended to larger teams in multiple work configurations consisting of many task *Requestors* and *Producers*. However, as we incorporate various synchronous and asynchronous interaction modes into larger teams, the complexity of the model grows exponentially. We also note that our model is consistent with other coordination models in the global software team literature. For example, one model suggests that actors need to communicate to make decisions that are mutually constraining and that this communication is affected by time separation (Herbsleb and Mockus 2003). Another model suggests that communication is the main mechanism through which informational coordination is achieved (Chaudhury *et al*. 1996).

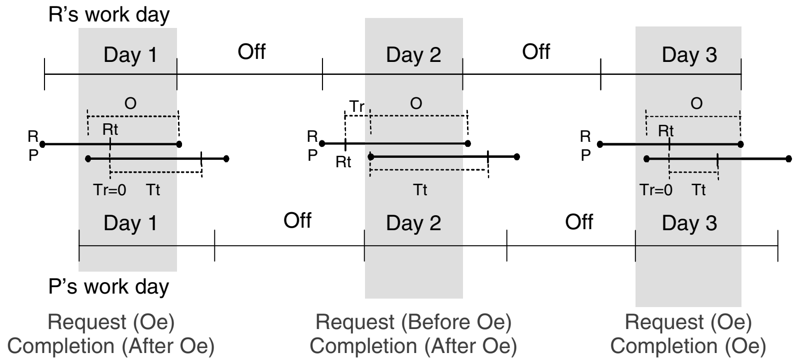

Figure 3. Graphical depiction of two actors with time-zone separation •Overlap time is depicted in yellow. R is the Requestor, P is the Producer. In Day 1, the request is made during the overlap period but the task is completed after overlapping hours; in Day 2, the request is made before overlapping hours and the task is completed after overlapping hours; in Day 3, the task is requested and completed during overlapping hours

4.1. Applications of Our Model

One of our main goals when we developed our model was to keep it as simple as possible, while retaining its explanatory power. The model involves a single collaboration act between two team members who have a sequential dependency, and it decomposes the total cost of carrying out this act into production and coordination. Coordination costs are further decomposed into four components: communication, delay, clarification, and rework costs. We argue that this model is very useful because it offers a fine level of granularity of coordination costs at the root of the collaboration process, which can help us understand coordination costs in more complex collaboration arrangements in which team members are separated by time. Parsimony is a widely accepted property for theoretical models (Rosenthal and Rosnow 1991), but it is particularly important for our model because things complicate rapidly as we add more members and time differences to the team. It is precisely this simplicity that makes our model useful to understand coordination costs in more complex team structures.

The power of the model resides in its ability to be adapted to more real conditions by changing parameters and by relaxing assumptions. These are some examples of possible expansions of the model: (a) •the presence of multiple synchronous and asynchronous communication tools can be modeled with a choice function based on the communication costs and expected communication quality payoff based on *Pu*; (b) larger teams can be modeled using network analytic methods in which each team member is represented as a network node with a particular *Cp* and *Cd*, and each collaborating dyad in the team is represented as a *Requestor–Producer* relationship; (c) different types of tasks can be modeled by manipulating the task duration *Tt*, the frequency of task request λ, the task equivocality (i.e. equivocal tasks are more uncertain, thus require more clarifications, thus a higher *Pu*), and the type of dependencies involved (i.e. sequential or reciprocal); and (d) multitasking can be represented by assigning priorities to tasks and additional delays to *Td* on the basis of these priorities. Table 4 illustrates how different components of our model help us understand the effect of time separation on coordination costs in a number of important GSD (global software development) practices.

5. DISCUSSION

Our study has several limitations. First, while we draw from three perspectives – practice, theory, and modeling – to provide a unified view of the coordination challenges in time-separated contexts, we only describe our model briefly. We have described our model in more detail in another paper, but the model still needs further development and empirical validation. Our model is based on simplifying assumptions, which we plan to relax as we develop it further. For example, we made no distinctions between the granularities of a requested task. More complex tasks that contain many subtasks would need to be modeled with

Table 4. Applications of our time-separation model

GSD practice	Total costs				
		Coordination			
	Production $Pc = \lambda CpTt$	Communication $Cc = Cl + 2\lambda Cm$	Delay $Dc = \lambda TdCd$	Clarification $Cf = Pu(Cc + Dc)$	Rework $Rc = PuPrRwPc$
Follow-the-Sun (FtS) • Time-zone separation is large with little to no time overlap. • 2 or 3 dyads • Task dependency is high by definition. • Task requests are close to the overlap period	• Production costs are only affected by the producer's daily production cost rate and the task duration. • Not affected by time separation.	• Costs will be low because communication is mostly asynchronous due to time-zone differences, which may increase *Pu* • More costly synchronous communication happens during brief overlap periods.	• Delay costs can be minimal if tasks are requested shortly before overlap time (i.e. *Td* is counted only during the requestor's working hours; it is zero otherwise) • Thus, well-programmed tasks are critical for FtS • Failure to make timely task requests can bring prohibitive delay (i.e. *Td* advances without production)	• Clarification costs are very high if task requirements are vague or uncertain or if communications are unclear (i.e. *Pu* is high), particularly if clarifications are requested during nonoverlap hours. • *Pu* is affected by the quality of the communication medium, by the equivocality of the task, and by how clearly the producer conveys task requirements. • As *Pu* increases, additional communication and delay costs are incurred in order to clarify messages.	• Rework costs can also be very high if requirements are found to be incorrect after development work has started. • Rework costs will be minimized when *Pu* is reduced with clear requirements and good communication tools; • *Pr* and *Rw* are reduced by incurring clarification costs early in the production cycle to detect incorrect requirements as soon as possible.

Offshore Outsourcing (OO) • Time-zone separation may be large or small • 2 dyads (in simple case). • Task dependency may be high or low.	• Companies outsource software development offshore because daily production cost rates *Cp* are low in low wage countries. • But coordination costs can be high.	• Higher quality communication tools will improve the media richness, thus lowering *Pu*, which will increase communication costs but will also reduce clarification and rework costs.	• Delay costs are low if overlap is small and task requests are batched and sent shortly before the overlap – *Td* is small if work can be done while the requestor sleeps. • In contrast, delay cost can be substantial if task request are made several hours before the overlap time. • Delay costs are high in same-time contexts because the delay *Td* is equal to the task completion time *Tt*	• Costs are reduced with longer overlaps because *Td* is no longer dependent on *Tt* but on how long it takes to get the clarification response from the task producer. • If the response does not arrive during the overlap period, the task producer has to wait a full day to get a response.	• Rework costs are higher when *Pu* increases. • Rework costs are affected by the producer's daily cost of production *Cp*. So, rework costs in OO can be small if *Cp* is small, thus, the incentive to work with countries where labor rates are low.
Large-Scale Global Software Development (LSGSD) • Multiple dyads with many task producers with varying degrees of dependencies • Time-zone separation may be large or small	• Large number of individual task producers in many locations, each with a different *Cp*. • Task requestors and producers may be grouped in locations in a number of different configurations.	• Same as OO • Communication costs will vary for each dyad depending on the locations involved and on the quality of the communication tools. • Owing to large number of people involved, effective communication and collaboration tools are critical.	• Same as OO • There are many task requestors with different *Cd's* in multiple locations. • Delay costs are lower with a distributed team configuration that maximizes dependencies within sites and minimizes dependencies between sites – i.e. low λ (e.g. coding in one site, testing in another; core development in one site, customization in another).	• Clarification costs are lower with effective asynchronous collaboration tools like configuration management systems. Besides change management, these tools can be used to describe issues and requirements, record errors, repair miscommunication, etc., all of which reduce *Pu*.	• Rework costs are lower with a lower *Pu* due to effective collaboration tools. • Rework costs can be reduced further by incurring clarification costs early in the task to reduce *Rw*. • On the other hand, if *Cp* is low for the producer's site, but *Cd* is high for the requestor, it may pay off to risk the possibility of rework to reduce clarification costs.

a sequence of communication events rather than a simple request and acknowledgement. We also assumed that actors communicate synchronously during overlapping periods and asynchronously otherwise. In reality, actors who wish to communicate during nonoverlapping hours have the choice of communicating asynchronously or of waiting until the overlapping period and then communicating synchronously, which can be modeled with a delay cost rate function dependent on the priority of the task that takes into account the additional cost of waiting against the expected gain in message clarity. We further assumed that face-to-face communication occurs instantaneously and at no cost. In reality, face-to-face meetings and preparing task-request messages can consume substantial productive time. This can be modeled by incorporating further time delays based on the task complexity, which will affect the message preparation time and the number of meeting participants, which creates production blocking (i.e. only one person can talk at a time). Nevertheless, the practical, theoretical, and modeling perspectives discussed in this paper underscore the differences between collaborations in software development that are purely separated by geographic distance from those that are also separated by time differences. We now discuss the overarching issues that emerged in this study.

5.1. Time Separation Means Reduced Overlap in Work Hours, not Time-zone Differences

Time separation boils down to the amount of overlapping work time in which the team can interact synchronously. AQ9 •This work-time overlap, not just because of time zones but also because of factors such as nonoverlapping weekend days and holidays, shifts, and different working schedules, can be reduced. An important feature of our model is that it purposely omits any reference to time zones and focuses more specifically on time separation. We represent this time separation in reverse, using a work-time overlap index (O'Leary and Cummings 2002), which can be used to model any form of time separation among team members.

5.2. Time Separation Leads Most Teams to Change their Work Norms

Specifically, individuals and teams adjust and shift their work hours to change work-time overlaps to suit their needs. Our model contains five cost components: production, communication, delay, clarification, and *rework costs*. If a given time-separation configuration is not cost-effective (e.g. due to time zones), rational actors will make decisions to change work schedules of some or all of its members to either increase time overlaps to reduce clarification and *rework costs* (e.g. create liaison roles) or reduce time overlap to reduce *delay costs* (e.g. shift work, *follow-the-sun*), provided that the timing of task requests can be programmed optimally.

5.3. Time Separation's Impact on Team Interaction Leads to Choices of Synchronous or Asynchronous Communication in a Number of Ways

In general, when team members are only separated by geographic distance, they have a choice of interaction mode. We recognize that there are times when one mode may be more effective than the other (e.g. send e-mail when a person is away from the desk), but because work hours fully overlap, there are more communication options. Teams separated by time have fewer choices on how to interact, and they often need to make choices between synchronous and asynchronous interaction tactics. Our model can be simulated under a number of different assumptions. For example, one simplifying assumption we made in a recent study (Espinosa *et al*. 2003) was that actors communicate synchronously during work-overlap hours and asynchronously otherwise. This assumption can be relaxed to model more realistic conditions. For example, if we assume that actors make rational choices, then an actor may either: (a) communicate asynchronously (e.g. e-mail) during overlapping hours because the message is very technical and it is better explained in writing, thus reducing the probability of unclear messages and reducing *vulnerability costs*; or (b) defer communication until hours overlap to communicate synchronously to discuss more equivocal matters over a richer medium (e.g. videoconference), thus reducing the probability of unclear messages and reducing *vulnerability costs*. These rational choices would involve actors making decisions on the basis of probabilities and trade-offs between *delay costs* and *vulnerability costs*.

5.4. Distance Separation is Symmetric – i.e. Distance (A,B) = Distance (B,A) – while Time Separation is Asymmetric

The type of overlap (i.e. at the beginning or end of one's workday) makes a difference in time-separated work but is not an issue in purely geographically dispersed contexts. While making task requests later in the day diminishes the benefits of overlap time, making late requests are somewhat more beneficial when the work-overlap time occurs at the end of one's day. Planning interactions and task work needs to take into account, when overlapping work hours occur. The main effect of this asymmetry is that the timing of a task request (or a task completion acknowledgement) really matters in time-separated contexts, whereas the timing does not matter in distance-only contexts. The simplified cost formulas we have presented in this article don't incorporate this asymmetry directly. However, the computation of time delay (*Td*), which affects two of the four coordination cost components (i.e. delay and clarification), is affected by this asymmetry. The effects of this asymmetry surfaced visibly in the model evaluations we conducted with simulated data (Espinosa *et al*. 2003, Espinosa and Carmel 2004).

5.5. In Time-separated Contexts, the Type of Time Separation Configuration Makes a Difference

While different distance separation arrangements matter in collaboration, teams that are not separated by time can still use a variety of synchronous communication tools (e.g. voiceconference, videoconference) and initiate instant interactions as needed. On the other hand, the more complex the time-separation configuration of a team, the more difficult it becomes to initiate or plan team interactions. Our model makes evident the cost trade-offs of different time-separation conditions and the manner in which they are affected by the nature of the task and the quality of the communication media available. Equivocal tasks (e.g. requirements engineering and design) that require more frequent interaction over rich media are more effective in work configurations with substantial work-time overlap among members so that *vulnerability costs* may be reduced (i.e. the probability of unclear messages is lower). On the other hand, less equivocal tasks (structured tasks, such as testing, and error fixing) may be better suited for *follow-the-sun* configurations that contain less overlapping work hours so that *delay costs* are reduced (i.e. assuming that the timing of task requests can be programmed optimally).

5.6. The Time Perspective Among Collaborators is the Same When they are Only Separated by Geographic Distance, but not When they are Separated by Time

When work times fully overlap, the time it takes to complete a task by someone else is equal to the time one has to wait for that task to be completed (i.e. $Tt = Td$). However, because of the asymmetric nature of time separation, when work hours do not overlap, the time it takes for one member to complete a task only affects the *Requestor's delay costs* if the waiting time occurs during the overlapping hours. If the work takes place during the *Requestor's* off-work hours, then that time does not affect *delay costs*. Conversely, if the task is requested before the *Producer* arrives to work, this produces extra delay in the *Requestor's* time, which is not perceptible to the *Producer*. This difference in time perspectives is often a source of misunderstanding and a lack of sensitivity to the other site's time constraints. This effect is captured in the model formulas in the computation of delay times (*Td*), which is measured from the *Producer's* perspective. Therefore, the timing of task activities is a critical issue in time-separated conditions but not when separated by distance only.

5.7. Vulnerability Costs Increase with Time Separation

Vulnerability costs – i.e. clarification plus rework costs – increase with time separation because of two reasons: (a) the timing of the interaction is affected by time differences, which is evident in our model by the interaction of the time variables (*Tt* and *Td*). Naturally, if miscommunication occurs frequently, time separation makes it difficult to interact frequently and spontaneously, thus introducing further delay; and (b) the choice of communication media is limited to the tactic employed (i.e. synchronous or asynchronous). In some cases, suboptimal communication media may be chosen, thus increasing the chance of miscommunication. *Vulnerability costs* are also affected by whether the team is colocated (or not) and by the amount of overlapping work time. It is not the same to have 80% overlapping work time between two sites as it is to have only 10%. The narrower the window for synchronous interaction, the fewer choices the team will have for synchronous communication tactics. This is also evident in our model in which the probability of unclear messages *Pu* is affected by the quality of the communication medium used and by the amount of work-time overlap available to repair miscommunication in a timely manner.

In conclusion, time separation has profound effects on the software process. Regardless of the software development method employed (e.g. waterfall, incremental, Unified Process, Extreme Programming), coordination is critical to the management of the software process, particularly, as the software size and the project team get larger (Brooks 1995). And, because software is a complex and equivocal task with intricate dependencies among multiple activities, communication is the key to accomplish coordination (March and Simon 1958, Thompson 1967) and to manage the software process dependencies effectively (Espinosa *et al*. 2001). The software process not only involves many developers making decisions and carrying out tasks individually but also involves subsequent coordination, which is necessary to integrate this individual work, resolve mutual constraints, and manage task dependencies. This coordination is not only necessary to produce software that meets requirements in a timely manner but it is also one of the most difficult and pervasive problems in the software process (Herbsleb and Mockus 2003). Time separation not only affects the timing of planned communication but it also affects team members' ability to interact frequently, informally, and spontaneously, which has an impact on the coordination of task activities in the software process (Kraut and Streeter 1995). In closing, we highlight our main argument that same-time and different-time collaboration contexts present different challenges for practice and research. Much of the research in global and geographically distributed teams does not distinguish distance separation from time separation. To avoid •confounds, we suggest that future empirical research in global software teams needs to either control for time differences within teams or be conducted with teams that are not separated by time. We expand on this theme by delineating the number of dyad interaction patterns that exist in time separation versus same-time teams. While in same-time contexts, there are only two possible collaboration modes, colocated or distributed; in different-time contexts, there are 16 possible collaboration modes depending on whether

- the collaboration is either colocated or distributed (2x);
- a member makes a task request during or outside the overlapping work hours (2x);
- the other member completes the task during or outside the overlapping work hours (2x); and
- the overlap occurs at the beginning or at the end of one's workday (2x).

This underscores the difference with pure distance-separated contexts, where time-related variables do not have a strong influence on coordination and *vulnerability costs*.

6. FUTURE RESEARCH

Having merged together theory, exploratory field research, and a basic model, we have defined a conceptual foundation for deeper research into time-separated coordination. We identify a number of research approaches for further study, which we discuss below:

6.1. Simulation Research

While we have provided preliminary validation of our model, our approach has been simple, using randomly drawn values from expected statistical distributions of variables; we believe that more formal and thorough simulation studies can provide further insights. Further simulation can both expand the model and relax some of our assumptions.

6.2. Experimental Research

Experimental studies can be used to hypothesize and test fine-grained aspects of our model and time-separated work in general. An experimental approach is likely to be designed around several time-overlap conditions, such as 0, 20, 50, 80 and 100%. Other variables may also be manipulated: task completion time (*Tt*) relative to the length of the workday; daily cost of delay (*Cd*); communication medium quality (i.e. affecting the probability of unclear messages); and amount of rework needed (*Rw*).

6.3.. Field Research

We see two steps in doing field work. The first is to continue and expand exploratory studies, interviewing, and surveying software developers in multiple organizations to: (a) identify effective design of work configurations in time-separated conditions; (b) develop a deeper understanding of the key issues that developers face in time-separated work arrangements; and (c) learn about how these teams cope with the challenges of time separation. Results of such a study can be used to refine our model. Second, we propose a case-study design at a single organization to explore relative *coordination costs* in different time-distance configurations. The organization may have either a single large team with members in multiple locations across different time zones or a number of smaller teams configured in a variety of configurations: (a) colocated team (i.e. the control condition); (b) dispersed sites across the same time zone; and (c) dispersed sites across different time zones. Such research can make use of three groups of data: interviews, survey, and system-derived data, possibly generated from the configuration management system.

REFERENCES

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

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

Brooks F. 1995. *The Mythical Man-Month: Essays on Software Engineering*. A. Wesley.

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

Carley KM, Lin ZA. 1997. A theoretical study of organizational performance under information distortion. *Management Science* **43**(7): 976–997.

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

Chaudhury A, Deng P-S, Rathnam S. 1996. A computational model of coordination. *IEEE Transactions on Systems, Man and Cybernetics* **26**(1): 132–141.

Chell E. 1998. Critical incident technique. In *Qualitative Methods and Analysis in Organizational Research*, Symon G, Cassell C (eds.). Sage Publications: Thousand Oaks, CA, 118–134.

Cramton CD. 2001. The mutual knowledge problem and its consequences for dispersed collaboration. *Organization Science* **12**(3): 346–371.

Dennis AR, Kinney ST. 1998. Testing media richness theory in the new media: The effects of cues, feedback, and task equivocality. *Information Systems Research* **9**(3): 256–274.

Espinosa JA. 2002. Shared mental models and coordination in large-scale, distributed software development. Doctoral dissertation, Carnegie Mellon University, Pittsburgh, UMI 3065743, ISBN 0-493-85125-9.

Espinosa JA,• Carmel E. 2003. Modeling coordination costs due to time separation in global software teams. *Global Software Development Workshop, International Conference on Software Engineering (ICSE)*. IEEE: Portland, OR. AQ11

Espinosa JA,• Carmel E. 2004. The effect of time separation on coordination costs in global software teams: A dyad model. *37th Hawaiian International Conference on System Sciences*. IEEE: Big Island, HI. AQ12

Espinosa JA,• Cummings JN, Wilson JM, Pearce BM. 2003. Team boundary issues across multiple global firms. *Journal of Management Information Systems* **19**(4). AQ13

Espinosa JA, Kraut RE, Lerch FJ, Slaughter SA, Herbsleb JD, Mockus A. 2001. *Shared Mental Models and Coordination in Large-Scale, Distributed Software Development*. International Conference in Information Systems: New Orleans, LA.

Espinosa JA, Kraut RE, Slaughter SA, Lerch FJ, Herbsleb JD, Mockus A. 2002. *Shared Mental Models, Familiarity, and Coordination: A Multi-Method Study of Distributed Software Teams*. International Conference in Information Systems: Barcelona, Spain.

Gittell JH. 2001. Supervisory span, relational coordination, and flight departure performance: A reassessment of postbureaucracy theory. *Academy of Management Journal* **12**(4): 468–483.

Grinter RE. 2000. Workflow systems: Occasions for success and failure. *Computer Supported Cooperative Work* **9**: 189–214.

Grinter RE,• Herbsleb JD, Perry DE. 1999. The geography of coordination: Dealing with distance in r&d work. *International ACM SIGGROUP Conference on Supporting Group Work (Group 99)*. ACM Press: Phoenix, AZ. AQ14

Gumpert DE. 2004. A New Tide of Outsourcing. Business Week.

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

Herbsleb JD, Mockus A. 2003. Formulation and preliminary test of an empirical theory of coordination in software engineering. *European Software Engineering Conference, Symposium on the Foundations of Software Engineering*. ACM Press: Helsinki, Finland•.

Herbsleb J, Mockus A, Finholt T, Grinter RE. 2001. An empirical study of global software development: Distance and speed. *23rd. International Conference on Software Engineering (ICSE)*. IEEE Computer Society Press: Toronto, Canada•.

Jehiel P. 1999. Information aggregation and communication in organizations. *Management Science* **45**(5): 659–669.

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

Klein JA, Kleinhanns A. 2003. Closing the time gap in virtual teams. In *Virtual Teams That Work: Creating Conditions for Virtual Team Effectiveness*, Gibson CB, Cohen SG (eds.). Jossey-Bass: San Francisco, CA, 381–399.

Koushik MV, Mookerjee VS. 1995. Modeling coordination in software construction – an analytical approach. *Information Systems Research* **6**(3): 220–254.

Kraut RE, Streeter LA. 1995. Coordination in software development. *Communications of the ACM* **38**(3): 69–81.

Lau D, Murnighan JK. 1998. Demographic diversity and faultlines: The compositional dynamics of organizational groups. *Academy of Management Review* **23**(2): 325–340.

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

Malone T, Crowston K. 1990. What is coordination theory and how can it help design cooperative work systems. *Computer Supported Collaborative Work*. ACM Press: Los Angeles, CA•.

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

Malone TW, Crowston K, Lee J, Pentland B, Dellarocas C, Wyner G, Quimby J, Osborn CS, Bernstein A, Herman G, Klein M. 1999. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science* **45**(3): 425–443.

March J, Simon H. 1958. *Organizations*. John Wiley & Sons•.

McDonough EF, Kahn K, Barczak G. 2001. An investigation of the use of global, virtual, and colocated new product development teams. *Journal of Product Innovation Management* **18**(2): 110–120.

O'Leary MB. 2001. Varieties of virtuality: Separate but not equally. FIU Workshop on Distributed Work and Virtuality, Miami, FL.

O'Leary MB, Cummings JN. 2002. The spatial, temporal, and configurational characteristics of geographic dispersion in teams. Presented at the Academy of Management Conference, Denver, CO.

Olson GM, Olson JS. 2000. Distance matters. *Human-Computer Interaction* **15**(1): 139–179.

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

Rosenthal R, Rosnow RL. 1991. *Essentials of Behavioral Research: Methods and Data Analysis*. McGraw-Hill: New York.

Teasley SD, Covi LA, Krishnan MS, Olson JS. 2002. Rapid software development through team collocation. *IEEE Transactions on Software Engineering* **28**(7): 671–683.

Thompson J.• 1967. *Organizations in Action*. McGraw-Hill.

Van den Bulte C, Moenaert R. 1998. The effects of r&d team co-location on communication patterns among r&d, marketing, and manufacturing. *Management Science* **44**(11): S1–S18.

van Fenema PC, Qureshi S. 2004. A phenomenological exploration of adaptation in a polycontextual work environment. *37th Hawaiian International Conference on System Sciences.*• IEEE: Big Island, HI.

VanDeVen AH, Delbecq LA, Koenig RJ. 1976. Determinants of coordination modes within organizations. *American Sociological Review* **41**(April): 322–338.

Waller MJ. 1999. The timing of adaptive group responses to nonroutine events. *Academy of Management Journal* **42**(2): 127–137.

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

AQ15 AQ16 AQ17 AQ18 AQ19 AQ20

QUERIES TO BE ANSWERED BY AUTHOR

IMPORTANT NOTE: Please mark your corrections and answers to these queries directly onto the proof at the relevant place. Do NOT mark your corrections on this query sheet.

Queries from the Copyeditor:

AQ1 Please clarify if this figure should be in color.
AQ2 Please clarify if this citation refers to 'Espinosa and Carmel 2003' or 'Espinosa *et al*. 2003'.
AQ3 Please clarify if this part of the sentence can be rephrased as ''we made interpretations of those events described that involved time separation''
AQ4 Please clarify whether the word ''those'' can be replaced by ''certain''.
AQ5 Please clarify if this sentence can be rephrased as ''we turn to coordination theories for information on the deeper understanding of the impact of time differences''.
AQ6 Please clarify if this abbreviation should be expanded in the first instance and if so please give the expansion.
AQ7 We have deleted the words ''in order'' from this sentence. Please clarify if it retains the intended meaning.
AQ8 May we rephrase this part of the sentence as 'the presence of multiple synchronous and asynchronous communication tools can be modeled by a choice function based on the communication costs . . . '?
AQ9 We have modified this sentence. Please confirm if it retains the intended meaning.
AQ10 Please clarify if ''confounds'' can replaced with ''confusion''.
AQ11 Please provide the page range for this reference.
AQ12 Please provide the page range for this reference.
AQ13 Please provide the page range for this reference.
AQ14 Please provide the page range for this reference.
AQ15 Please provide the page range for this reference.
AQ16 Please provide the page range for this reference.
AQ17 Please provide the page range for this reference.
AQ18 Please provide the place of publication for this reference.
AQ19 Please provide the place of publication for this reference.
AQ20 Please provide the page range for this reference.