

# Using ICT to Avoid Communication: How ICT's *really* enable distributed software development<sup>1</sup>

Kannan Srikanth  
Ph.D. Student  
Strategic and International Management Area  
London Business School  
Sussex Place, Regent's Park  
London NW1 4SA  
[ksrikanth@london.edu](mailto:ksrikanth@london.edu)  
Phone: +44-(0)20-7000-8762  
FAX: +44-(0)20-7000-8701

Phanish Puranam  
Asst. Professor  
Strategic and International Management Area  
London Business School  
Sussex Place, Regent's Park  
London NW1 4SA  
[ppuranam@london.edu](mailto:ppuranam@london.edu)  
Phone: +44-(0)20-7000-8742  
FAX: +44-(0)20-7000-8701

---

<sup>1</sup> We acknowledge funding from the Leverhulme Trust Digital Divide Programme at London Business School. We are also grateful for useful suggestions over the course of this project from Matthew Bidwell, Stephan Billinger, Isabel Fernandez-Mateo, Gerry George, Reddi Kotha, Ruiming Lin, Felipe Monteiro, Peter Moran, Yiorgos Mylonadis, Jim Oldroyd, Madan Pillutla, Huggy Rao, Olav Sorenson, and Bruce Weber.

# Using ICT to Avoid Communication: How ICT's *really* enable distributed software development

## ABSTRACT

The successful execution of complex interdependent work in a distributed fashion- such as offshore software development- is an anomaly, given the well known inadequacies of ICT mediated communication vis-à-vis face to face communication. In this paper, we draw on qualitative data from 60 distributed and collocated software services delivery projects to understand how ICT's are actually used in distributed software development. We find that in these projects, ICT based tools are typically not used as channels of direct communication between locations. Instead ICT tools are used to avoid the need for direct communication by creating common ground across locations and thereby enabling tacit coordination.

### Key Words

Coordination, ICT tools, Global delivery, Distributed organizations, Offshoring

## INTRODUCTION

The economic importance of the geographic dispersion of value chains- often referred to as offshoring - can be gauged by projections of the scale of dispersed work. It is estimated that up to 20% of all employment of the OECD countries in North America and Western Europe could be affected by offshoring<sup>i</sup>. Another report estimates that 16% of all work in the world's IT-services industry is already carried out remotely<sup>ii</sup>. By 2009, offshore services are expected to reach 20% or more of the US \$760 Billion global spending on IT services<sup>iii</sup>. The former editor of MIS Quarterly, in his recount of the greatest developments in our field since Y2K counts offshoring of IT services as one of the most significant developments for practitioners and academics (Weber, 2004). And yet compared to what we know about the impact of Information and Communication Technologies (ICT) on the unbundling of the corporation (Hagel and Singer, 1999; Evans and Wurster, 1999) we know relatively little about the specific role that ICT's play in allowing the offshoring of services<sup>iv</sup>.

Even an initial examination of the role that ICT's play in enabling offshoring reveals the existence of a paradox of sorts. As channels of communication, ICT's continue to fare poorly compared to physical collocation in terms of media richness as well as the ability to provide social presence (Olson et al 2002; Kraut et al, 2002). These results, at a minimum, coexist with some tension with the staggering success of the offshore services industry (Ethiraj et al, 2005; Athreye, 2005). Consider specifically the case of software development, which involves the coordination of complex and changing patterns of interdependence (Cataldo et al, 2006; Kraut & Streeter, 1995; Adler, 2005). Code architectures are often revised mid-project, and change requests are a constant feature of software project management. Indeed, recent trends have begun to emphasize the value of "radical collocation" in software development, where programmers sit cheek by jowl (Olson and Olson, 2000; Olson et al, 2002). If ICT based communication is a poor substitute for

collocated activity, and software development requires extensive coordination, how does one account for the runaway success of offshore software development?

One obvious, but in our view unconvincing, explanation is that labour arbitrage compensates for coordination losses. Thus, cost savings from using low wage programmers from countries such as India could compensate for the sub-optimal coordination achieved through ICT's instead of collocated development. However, as software experts know, it is difficult to trade-off cost against coordination in the creation of software. Poorly coordinated projects result in errors, re-work and schedule slippages (Kraut & Streeter, 1995; Cataldo et al, 2006), and reliability and timely development are often non-negotiable aspects of software development projects

A second (and more intriguing) proposition is that the offshore software development model has evolved features that compensate for the inadequacies of ICT's as communication media. We investigate this proposition by drawing on qualitative data from 18 collocated and 42 distributed software development and maintenance projects across two large well known software development firms. Through an analysis of interview and archival data, we attempted to understand exactly how ICT's are used (despite their inability to provide the richness characteristic of face to face communication) to coordinate distributed software development.

Our results point to the surprising conclusion that the primary role of ICT's in the distributed projects we studied was not as a substitute for collocated communication, but instead to enable coordination by updating and maintaining "common ground" across locations. Common ground between two people is defined as "the sum of their mutual, common or joint knowledge, beliefs and suppositions" (Clark, 1996; p93). Common ground is a more tractable version of the economic notion of common knowledge<sup>2</sup>.

---

<sup>2</sup> The concept of common ground is closely related to the economic concept of common knowledge. Common ground defined as iterated propositions (Clark, 1996, p95) is exactly the same as common knowledge. However, common ground can also be defined from a shared basis or as reflexive. Clark suggests that common ground as iterated propositions is psychologically infeasible (p96). Aumann and Brandenburger (1995) similarly analytically proved that "mutual knowledge" which is identical to common ground reflexive is adequate for the economic theories of coordination games and the more restrictive definition of common knowledge is not necessary. i.e., instead of the "I know you know I know you know..." ad-inifinitum, their result shows that just two iterations of I know you know, which they call mutual knowledge are sufficient.

In the context of distributed software development, common ground refers to the joint knowledge across locations of who is working on what, what stage they are at, who is interdependent with whom, and information regarding the abilities and constraints facing participants in each location. This common ground a) made communication less necessary and b) transformed even frugal communication, when it occurred, into rich meanings that allowed for coordinated adaptation across locations. Thus ICT's may enable the coordination of distributed activity not through providing channels for rich communication, but by making such communication unnecessary.

Our research complements the growing body of work that has examined the use of ICT mediated communication in laboratory settings (see reviews by Kiesler and Cummings, 2002; Walther, 2002; McLeod, 1996), and the literature on coordination mechanisms in organizations (March and Simon, 1958; Nadler and Tushman, 1998; Grandori, 1997; Rivkin and Siggelkow, 2003). Specifically, our results underscore the points that common ground can support tacit coordination – coordination that occurs in the absence of any communication (Schelling, 1962; see Camerer, 2003 for a comprehensive review of this literature). This is in contrast to explicit coordination which relies either on anticipatory planning or on ongoing communication (March and Simon, 1958). Thus, our study indicates that the common ground generating capabilities of ICT's can compensate for their weakness as direct communication channels.

The rest of this paper is organized as follows: First we provide a brief review of the existing literature on the use of Information and Communication Technology tools in facilitating remote work. Next we explain our qualitative data collection methodology and analysis procedures while noting why these are suitable for the current research. Then we present our findings regarding how ICT tools are used in the field to coordinate remote work. Finally, we discuss the contributions this paper makes to research and practice.

## **BACKGROUND: PRIOR LITERATURE ON ICT AND REMOTE WORK**

A simple baseline explanation for the ability of ICT's to enable remote coordination despite their limitations, is that distributed work is partitioned in a manner that minimizes interdependence across locations. In other words, work can be redesigned to fit the situation (Hackman and Oldham, 1990; Wageman, 1995) – in this case minimize interdependence across locations. This strategy of modularization or a pre-planned partitioning of the work into chunks with minimal interdependence between them is well known in the literature on organization design (March and Simon, 1958; Tushman and Nadler, 1977; Simon, 1962; Baldwin and Clark, 2000), as it is in the world of software development (Parnas et al, 1981.). A system architect can partition up-front the code architecture into modules such that each developer working on a module does not need to worry about the decisions of any other developer working on another module. In effect, the architect has then “planned” the modules such that the standardized interfaces between modules fully capture all the information about it that is necessary for integrating it with other modules. There is indeed some evidence that open source software systems are coordinated among thousands of voluntary participants mainly through the use of modular code architectures (Moon and Sproull, 2002; Mockus et al, 2002).

However, there are often practical constraints to achieving a modular architecture. First individuals are boundedly rational and may not understand the systems and the nature of interdependencies in enough detail to achieve a suitably modular solution (Simon, 1962). Such limitations may be due to both cognitive limitations as well as lack of information (Ethiraj and Levinthal, 2004). Second, planned coordination strategies, such as modularity, standards and procedures all require the ability to anticipate the nature of the required software modules and functions. In dynamic situations where the underlying patterns of interdependence are unknown or changing, such mechanisms are insufficient. For instance

changing functions and requirements and practical constraints such as legacy or heritage systems limit the use of modular code architectures to minimize interdependence across locations in distributed software development, though they do not rule it out.

Traditionally, ongoing communication is the de facto coordination strategy identified when planned coordination strategies (such as modularity) are not feasible. March and Simon, (1958) suggest that “feedback” – ongoing two-way communication - is used when it is impossible to plan effectively how to coordinate (through modular interfaces, for instance). When interdependence cuts across locations, whatever communication occurs must necessarily be by means of ICT tools such as email, telephone, pager, videoconference, computer conference etc. Indeed the “null hypothesis” in the minds of most practitioners as well as researchers regarding what enables remote coordination is “ICT enabled feedback”: i.e., remote teams make use of the increasing sophistication and availability of ICT enabled communication tools to coordinate their activities.

A large literature examines whether ICT mediated communication can approximate face-to-face communication (De Meyer, 1991; Kraut et al, 1988; McGuire et al, 1987; Siegel et al, 1986; Walther, 2002). These include both lab and field studies. In lab studies, participants make a decision or solve a problem, either face-to-face or using ICT mediated communication. Field studies tracked how real-world teams tackled problems using available media tools. The essence of the findings from these studies is that all forms of ICT mediated communication perform poorly vis-à-vis face to face communication in terms of “bandwidth” – the ability to convey non-verbal and visual cues- (Short et al, 1976; Daft & Lengel, 1984; 1986) and synchrony – the ability to provide and receive immediate feedback – (Kraut et al, 2002; see Olson et al, 2002; see Kiesler and Cummings, 2002; Kraut et al, 2002, and McLeod, 1996 for reviews).

While limitations of low synchrony tools (such as email) that do not allow linguistic context to be well preserved - it is not always clear what question was asked and how the answer relates to it (Kraut et al, 2002) – are well known, recent research suggests that even media such as telephones or computer mediated voice conferencing that we think of as synchronous may not be “as synchronous” as face-to-face communication (Kraut et al 2002). For instance, communication media that introduce even small delays can make comprehension substantially more difficult to accomplish (Krauss & Bricker, 1966). This is because it is more difficult to build on and utilize common ground in such jerky conversations (Clark, 1996; Clark & Brennan, 1990). Communication using media that generates delays is considered less natural, and typically transfers less information, and is terminated sooner (O’Conaill, Whittaker, Wilbur, 1993). In sum, media such as email and telephone are useful to effectively convey certain types of information that can be unambiguously interpreted or have very little equivocality (Daft & Lengel, 1986). They are less suitable to convey information that is harder to interpret or is highly equivocal, such as for tasks involving consensus forming, negotiation, or joint exploration of an unstructured problem, which arguably characterize a significant portion of software engineering activities.

The limitations of low bandwidth tools are of course well known. However, recent research suggests that even the highest bandwidth ICT tool available today, video-conferencing, may also have significant limitations as a communication medium because it neither provides high-fidelity interactivity such as in face-to-face communication, nor the social benefits of collocation (Doherty-Sneddon et al, 1997; Heath & Luff, 1991; O’Conaill et al, 1993; Whittaker, 1995).

In sum, the remarkable success of the global offshore software services industry and the ability that companies in this sector have developed to deliver fairly complex projects from remote locations pose a puzzle to our understanding of distributed coordination. The

known shortcomings in ICT mediated communication and possible limits to achieving modularization across locations appear to have been surmounted in the field – though we do not know how. We therefore investigated how ICT tools are actually used in the field and the mechanisms organizations use to compensate for their inadequacies to coordinate software projects successfully across distances of thousands of miles.

## **METHODOLOGY**

Data collection on the importance and use of ICT tools in distributed software development was part of a larger project on offshore software services delivery. We interviewed project managers for 60 projects undertaken by two large software services firms, one headquartered in the US and the other in India. The US firm, that we call “Integrator” is a well known software services provider with extensive experience in delivering large globally distributed projects for its multinational clients, and has a significant presence in all continents. The Indian firm, that we call “Process Master” is considered a pioneer in offshore software services and has been extremely successful in the past decade and has made a reputation for itself in delivering global solutions. This firm also currently enjoys a significant presence in all continents. We specifically selected these two firms, since in the industry it is known that one of these firms (ProcessMaster) is better than the other at offshore delivery, while the other firm (Integrator) is relatively new to offshore delivery. However, both increasingly compete for similar projects, thus making comparisons between them potentially insightful.

### **Case Selection**

Selection of cases for this study followed theoretical sampling (Eisenhardt, 1989; Yin, 1994) and is displayed in Figure 1. Specifically, our sponsors within these companies helped us to identify project managers who had experience managing projects in at least two of the cells in Figure 1. The managers were asked to think of two specific projects, one for each cell,

and asked to draw comparisons between them on the coordination mechanisms used and their perceived success. We included collocated projects in the sampling, since we were keen on understanding the different coordination problems that remote work posed and the ICT tools that were used as a consequence to mitigate these problems.

INSERT FIGURE 1 HERE

We studied both development and maintenance projects. Development projects are those where a client organization hires a vendor to create new software to provide some specified functionality. Development projects typically involve novel tasks and unique situation based requirements and may be more prone to the need for collocation since the client and vendor have to negotiate to arrive at a joint understanding of what the requirements are and how best to satisfy them. Maintenance projects on the other hand involve ongoing support and upkeep of pre-existing software and systems. Maintenance projects may not require as much creative effort in design as development projects, but they do involve complex coordination events around quickly resolving problems with systems that are critical to the organization. Apart from ensuring variation in the degree of collocation and degree of collaboration between client and vendor, projects were sampled such that there was variation in technological complexity and project size.

### **Data Collection and Analysis**

Data collection and analysis followed grounded theory building techniques (Strauss and Corbin, 1997; Miles and Huberman, 1984). We first contacted managers in these two organizations with whom we had previous ties, explained that we are interested in studying coordination techniques in software projects and requested them to recommend and arrange meetings with appropriate project managers. The interviews typically lasted between 60 and 180 minutes. Most of these interviews were recorded (with permission from the respondent), and we also took extensive notes during the interview. In many instances the respondents

drew diagrams and charts on a white board to explain concepts around which clarifying question were asked, and also provided us with copies of documents, slides and templates that were used in their firm.

The study was conducted in the following phases. In the first phase, we conducted preliminary interviews with four project managers in order to identify the issues and phases that we should focus on in the main data collection phase. This data was used to familiarize us with the setting and the jargon. In this phase we asked open-ended questions regarding coordination issues in software projects. Based on this pilot study, we prepared a list of questions to act as a guide to semi-structured interviews with software project managers.

In the second phase of data collection, we interviewed a set of managers who could talk to us about their experiences in specific (named) projects that they were involved in across at least 2 cells in our design. These managers were identified by our “sponsors” in the organizations and were contacted with a short statement of the purpose of our research and a list of the type of questions we will be asking in the interviews to help the managers prepare and also assemble secondary material before our arrival.

The interviews in this phase followed the interview protocol that was prepared. Apart from recording these interviews with permission, the researchers took notes, which were typed up within 24 hours of the interview along with any field observations. Some project managers were contacted again, either in person, on the phone or by email to improve our understanding and resolve any inconsistencies.

In these interviews we specifically asked the managers to list the coordination tools they used in this project, what was available in each of the locations that the project was conducted from, how often were they used, and how important they were perceived to be in determining the success of the project. In most cases, we asked for details about why a particular tool was important and how the project could be coordinated had that tool been unavailable.

In the third phase the analysis of the evidence proceeded by iteration, and proceeded in tandem with the second phase. Both researchers read the interview transcripts and discussed emerging ideas and themes that were subsequently incorporated in the additional interviews to gain more data. Specifically, we followed the flip-flop technique and the far-out comparison technique (Strauss and Corbin, 1996) to enhance theoretical sensitivity. As theoretical categories emerged, we incorporated them in subsequent interviews to achieve greater understanding of these concepts. The objective was to understand the properties of particular tools, the specific role they played in the project and the coordination mechanisms that the tool embodied. We were also sensitive to the importance of different tools in different phases of the project.

We were also able to collect performance data for 80% of the projects, either from the manager whom we interviewed (for 60% of the projects) or from a supervisor (for the rest 20%). The performance data collected include objective measures on budget and schedule slippage, as well as subjective measures of vendor and client satisfaction with the project (on a scale of 1-7). In an exploratory factor analysis, all the four items loaded on a single factor that explains over 80% of the variance.

## **FINDINGS**

### **Project Profiles**

In our sample of 60 projects, 42 projects were distributed across two or more locations and 18 projects were conducted wholly from a single location. We first compare distributed projects to collocated projects in terms of size and complexity, and then describe distributed projects in terms of their level of interdependence between locations. Table 1 reports how we measured concepts such as size, complexity and interdependence.

Table 2 describes distributed and collocated projects in terms of size and complexity. We remained faithful to the managers' description in terms of size and complexity. Large

projects typically had more than 50 FTE working on them. Complex projects typically involved leading-edge technologies and/or business domains in which the vendor firm had little experience. Perhaps unsurprisingly, large and complex projects tend to be overwhelmingly organized in a distributed fashion. While our projects were not sampled to represent the population of projects but rather based on theoretical sampling (Yin, 1994; Eisenhardt, 1989), our discussions with managers in both firms indicate that it is typical for large complex projects to be organized this way. As one manager explained,

“We have no other alternative but to use the distributed model for such [large and complex] projects. The developers with the required skills are not always available in a single location, and they do not always relocate. To have a large number of developers travelling will blow our budget. There is no other alternative to using distributed models for these projects”.

Interestingly, in our sample, 5 of the large collocated projects were so organized because of explicit client requirement (and a commitment from them to bear the additional cost). The managers of these projects also said that except in one case, these projects could be organized in a distributed fashion had the client permitted it. Small projects were equally likely to be distributed or collocated while medium sized projects were twice as likely to be distributed as collocated regardless of the complexity of the project.

INSERT TABLE 1, TABLE 2 HERE

Of the 42 projects in our sample that were distributed between two or more locations, our informants described 39 projects as having high interdependence between locations. All the 26 projects that were described as large complex and distributed also had high interdependence between locations. The inability to partition work such that there is little interdependence across locations mainly derives from two considerations: the need to interface with legacy systems and the unavoidable changes to requirements since they cannot be fully anticipated at the beginning of the project. For example, one manager commented:

“For this project, there was a dramatic change in the architecture midway through the project. Certain technical assumptions were made that drove the initial work

on the project. However, in the design phase the team found that the assumptions were wrong. This led to a complete re-think on the architecture of the system. However, the team structure itself or the distribution of work among locations did not change.”

Another manager spoke to us about a project with the need to customize and maintain 3<sup>rd</sup> party applications over whose architecture they had no control. Due to cost cutting initiatives from the client, they had to shift some work offshore. Another manager spoke about a project designed to take into account the time difference between locations, where the (same) piece of work shifted seamlessly between USA and India. Thus code architecture is typically not modular across locations.

Table 3 describes distributed and collocated projects in terms of their performance. Contrary to our expectations primed by prior research, in our sample of cases, distributed projects seem to be as successful as collocated projects. Of the 26 large and complex projects that were organized in the distributed model, managers reported that 6 had very poor performance, while of the 7 large collocated projects, 2 had very poor performance. Among the medium sized projects, 1 out of the 13 distributed projects and 1 out of the 7 collocated projects reported very poor performance, while none of the small projects, distributed or collocated, reported poor performance<sup>v</sup>. Our confidence in these comparative assessments is bolstered by the fact that each of our interviewees discussed cases in at least two cells in Figure 1. Thus in our data, the distributed and collocated projects are similar to each other in terms of complexity and performance, while larger projects were more often organized in a distributed fashion.

INSERT TABLE 3 HERE

Given that (a) geographically distributed projects in our study were characterized by high levels of interdependence between locations and (b) yet seemed as successful as collocated projects, we next set about to investigate the role that ICT tools played in enabling coordination across distance in these projects.

## **The (Non) Usage of Rich Media ICT tools**

We define ICT based communication as communication across locations that is facilitated by technologies such as telephone, Email, Instant Messenger (IM), Desktop sharing or electronic whiteboard facilities such as NET Meeting or Live Meeting, Videoconference, and Web Cams (Kraut et al, 2002). These ICT tools could be classified on the basis of the richness of communication they enable as shown in Figure 2. Most researchers hold face-to-face communication as the gold-standard for media rich interaction. Under conditions of low analyzability and low familiarity between interacting individuals, such as those in different locations, tools that provide high visual and social cues (or high bandwidth tools) are considered very important (Rice, 1993; Kraut et al, 2002).

INSERT FIGURE 2 HERE

In offshore software services, given the constraints on face to face communication due to geographic dispersion, we expected that the usage of rich media ICT's (such as Videoconferencing, computer conferencing, NET meeting, Live meeting) would be widespread, in order to substitute, however imperfectly, for face-to-face communication. The importance of videoconferencing has been documented in other distributed technology settings. For example, Friedman (2005) describes the collaborative work between Boeing in the US and the firms it has outsourced work to in Russia in airplane designs.

“Boeing has set up a 24-hour work day where they just pass their designs back and forth from Moscow to America ...There are videoconferencing facilities on every floor of Boeing's Moscow office, so engineers don't have to rely on email when they have a problem to solve with their American counterparts. *They can have a face-to-face conversation*” (p 195). [our emphasis]

We found that some of the software projects we studied had set up 24 hour work days that rivalled the complexity of this work. However, in our sample of 42 distributed projects, only 9 used any kind of rich ICT based communication tools to aid software development or maintenance (Figure 2 also indicates the number of projects that used each set of tools). Of

these nine projects, seven used NET Meeting, one used Live Meeting in addition to NET Meeting, and only two projects used videoconferencing. Interestingly, of the 9 projects that used rich media, 8 were from Integrator, only 1 from Process Master.

Even more surprising, the low usage could not have been due to the lack of access or familiarity to such tools. NET Meeting and Live Meeting are readily available and arguably, IT professionals are the ones most familiar and comfortable using this technology. As two managers from Integrator told us about a project that had taken place in 2001-2002:

“.....NET meeting and IM were not yet popular – the technologies were *available*, it was just that people were not using them. We did not even use conference calls”.

Even in contemporary projects, these tools were infrequently used.

“So [coordination is by] day by day communication in some way, usually email and phone, we do not use NET Meeting etc ... it’s not necessary to have so complicated tools to interact”.

For video conferencing, data bandwidth could not have been the constraint, because these companies have high capacity leased lines for carrying data and the cost of the equipment pales in comparison to the budgets and the complexity of some of these projects. We did some of our interviews with Process Master at their offshore locations, and many of the facilities where we conducted interviews were equipped with video conferencing equipment. However, they mainly used these facilities for internal meetings, sales presentations, and impressing potential clients with the technology available at their disposal, but said that they never used them for project related discussions. Of over 20 projects in our sample, Process Master used rich media in only one project for software development.

Thus, distributed projects in offshore software services delivery typically did not use rich media tools such as videoconferencing or computer conferencing facilities, even when such facilities were readily available, relying instead on lean communication tools such as telephone and email. Even the use of Instant Messenger was rare, though our respondents typically agreed that it was a very convenient medium because it allowed multi-tasking (i.e.,

answering something while on the telephone) and had lower psychological cost (“you can ignore an IM when it comes at an inconvenient time, it is very hard to ignore a ringing telephone”). Only 11 projects in our sample used other low bandwidth tools such as Instant Messaging. Of these 11, 10 were from Integrator, where the firm had invested in creating a proprietary messaging system to be used within its firewall. Interestingly, of these 11 projects, 7 also used NET Meeting or other rich media.

Instead, the media of choice for conducting the bulk of coordination across locations in offshore software services delivery is email, supplemented by scheduled telephone conversations. Why do such relatively poor media play a dominant role in coordination, even when relatively richer media are available? When asked about the non-use of richer media, and their reliance instead on poorer media such as email and telephones, most managers suggested, intriguingly, that “those were enough”. There are also practical constraints unconnected to cost or availability to using richer media in offshore software services. High synchrony media such as telephone and instant messenger, as well as some of the high bandwidth media we considered earlier, such as videoconferencing and computer conferencing require co-presence, i.e., they require that all parties to the communication be available simultaneously. This is usually not possible in offshore software services where teams work in different time zones.

For example, those working in India and the US Central Standard Time are 10.5 hours apart and have no overlap on a 8:00 – 17:00 work day. In our sample, one project required that the offshore office work nights, at the same time zone as the US (for Integrator), while two others requested an extended day to have one-two hours overlap (one each in Integrator and Process Master). The vast majority of the projects worked on a 24-hour basis with no overlap in time zones, and special arrangements were made for co-presence, such as pre-scheduling telephone calls. This limited the usefulness of telephones and instant messengers

as tools to achieve coordination by allowing “spontaneous” communication. This limitation is especially relevant for Instant Messaging, since the greatest utility of that technology is the perceived spontaneity with which communication can take place. And surprisingly, even when special arrangements were made, they were not always used. One manager told us:

We designated a person to be “on-call” in each location. They carried a mobile phone and were required to be available to answer any questions that might arise during the course of work in another location. However, this resource was hardly used.

However, the managers did uniformly agree that both email and telephone calls were extremely useful to clarify information and check the status of each team. Tele-conferences were routinely scheduled to discuss project related matters. Interestingly, both the timing and the content of the conversations on the telephone especially were mainly scripted. As a manager told us:

“The developers [across locations] just did not communicate whenever they have doubts or problems – at certain pre-specified milestones, according to process, they have to share certain documents and communicate. The communication at this meeting is not ad-hoc; they have to talk about certain things.”

These emails and teleconferences involve information transfer in the form of specific questions requiring specific answers, rather than unstructured problem solving. A manager recounted how tele-conferences were used in her project:

The London team will send a completed TDS (a document) offshore. The offshore team will refer to all the documents in the repositories and their own knowledge and exhaustively complete the TDS including all details. This will be emailed to the London team. In the tele-conference very specific questions regarding the document will be raised and answered or become an action item for somebody to answer.

Thus the communication that does occur through email and scheduled phone calls between locations in the case of offshore software development tends to be highly structured - to obtain information such as clarifications on tasks, asking where a piece of information resides, keep specific members up-to-date etc. How could this “be enough” to coordinate the

significant and changing patterns of interdependence typically found in software development (Kraut & Streeter, 1995; Cataldo et al, 2006; Herbsleb & Mockus, 2003)?

### **Using ICT's to build Common Ground in Distributed Software Development**

Our analysis suggests that relatively poor communication media like emails and scheduled phone conversations “were enough” because of the presence of another class of ICT based tools which directly enhanced common ground across locations, enabling tacit coordination. Specifically, there were three classes of ICT tools that achieved this- shared code repositories, configuration management tools and workflow management tools. These tools enhanced common ground between locations by making the latest information regarding the specifications, code, and bugs in the project available to all developers across all locations simultaneously. As a consequence, engineers sitting halfway across the world were able to adjust their tasks - such as coding strategies and bug fixes to that of their counterparts in other locations – with limited communication. Creating this shared view of work throughout the project allowed developers to understand their role in the system – how they related to everyone else, and helped them to execute their tasks by taking others work into consideration. These tools also prevented them in most cases from making coordination mistakes that would have ripple effects on other's work.

Managers for all the 60 projects, whether collocated or distributed cited the importance of these three kinds of tools (code repositories, configuration management tools and work flow management), though most agreed that they were even more important in distributed projects than in collocated projects. The evidence provided was both positive and negative – how having them helped coordination and how not having them resulted in poor coordination. We describe how each worked in more detail below.

**Shared Code Repositories:** All managers emphasized the importance of having shared repositories for all the artefacts produced in the project. A shared code repository

provided access to all project members with the latest versions of the code and the attendant documentation. It also provided members with a view of all the requirements and the history of the changes that a particular piece of code went through and the reasons for it, as well as any future modifications that should be expected. Most of this information would never be used by any given programmer; yet as one manager noted, having this common view of the documentation related to the project available for those few cases when it was necessary made it worthwhile:

We have a twenty-eighty (onsite-offshore) split actually, a lot of investigation had to be done; the system was opened up to us, okay, so even though they were able to talk to the customers and we were sitting here, half of us were sitting here and we could look at the code, we could look at the communication that had gone on and turn it over to make certain decisions. The entire system was opened up to us ... so the documentation helped us get an immense understanding of the system [offshore].

This manager suggests that offshore access to extensive documentation and code about this system enabled offshore developers to understand the behavior of their on-site counterparts, and make decisions regarding what should be done offshore that dovetails with the action that occurs onsite. The coordination problems caused by the lack of current information in shared repositories were noted by another manager:

“[Common repositories] helps everybody to know where they are. However, suddenly the USA team stopped using the repository and that caused a lot of problems. We did not know what they were doing. Tools were used, but not fully, sometimes documents were stored in local drives, mainly because they [USA team] did not want to spend the effort training to use these tools”.

This manager told us that this lack of common ground between sites had a significant negative effect on this project's outcomes, especially due to the inability to define requirements in each location that were aligned with the decisions taken in other locations, as well as mismatched code that did not integrate well.

**Shared Configuration Management Tools:** The next set of tools that the managers for every single project in our sample mentioned were configuration management tools.

Examples include source control and source code revision managers such as Polytron Version Control System (PVCS), software configuration management, configuration builders etc. Configuration management enables developers in all locations to (a) always access the latest code and functionality from other developers that they might need for their own work (b) prevent developers from inadvertently changing or removing the work of another developer by accessing the same module at the same time, and (c) embody the structure of the design by tracing the modules and the relations between them and making that structure explicit – so developers have a very good idea of who/what they are interdependent with.

In general, these tools achieve coordination by making it very hard for the developers to make genuine mistakes such as modifying someone else's code inadvertently or giving the same name to two programs (which is not a trivial problem in a system that contains thousands of programs). They also help in correcting errors quickly when they occur since they trace the dependencies between one module and another.

In principle, shared repositories and configuration management tools function similarly. The key distinction is that shared repositories are more current, are more extensive and have more “future” related information. Shared repositories are more important in the beginning and the main development stages of the project while configuration management tools are more important at the end of the project especially during change management.

**Shared Workflow and Information Management Tools:** The third class of ICT tools that build common ground are automated tools that message the developers with common information required in the coding, testing and change management phases of the project. These tools help achieve coordination by creating a shared awareness among the developers regarding who is working on what and who are the relevant participants in any decision.

While the previous two classes of tools we discussed were “pull” in the sense that the developer should seek out the information available in a repository, automated messaging

tools on the other hand are more “push” – where information is sent to an appropriate developer by others in the project regarding issues that the developers must be concerned with. Typically, these take the form of change request notes, or information regarding changes to their modules that someone else is performing. In the words of one manager:

We use a client developed in-house tracking tool called “X” to track bugs and modifications in the software. When the testing suite detects a bug it creates a ticket in X that automatically is assigned to the current all code owner but also sent to others who have had recent contact with the code [i.e., those modified that piece of code or those coding closely related systems and access this functionality]. This is absolutely helpful because this is a very old legacy system that has been in use for over 20 years throughout the [MNC] client firm and many modifications by many different vendors are happening on it from several locations. We don’t even know who some of them are and what they are doing with it. But because of “X” we know we have to interact with them in resolving this problem.

This tool is similar to email – but then it has key additional features to make it very useful for creating a shared understanding of the task requirements and decision participants: it provides relevant information in a format that is readily accessed, is secure and most important, and is traceable from anywhere in the project. These tools typically push this information to all developers who have current interest in the code, which prevents accidental omitting of key individuals as often occurs in emails. Finally, they are tracked by project management, easy traceability is a part of their design - this forces resolutions and avoids “things getting under the radar”. For example, Armstrong & Cole (2002) in their analysis of distributed projects found that several coordination (as well as motivation) problems occurred due to the accidental omission of key personnel in distant sites from crucial emails. These automated messaging tools go far in eliminating this problem. Accessibility and ease of use are key features of these tools as another manager explained below:

In this project we used a web tool for all those sort of like ... problem resolutions, whether it be code, analysis, whatever, we used the same tool throughout that you could actually ... anybody who has a web tool could log their problem, you could get a status update, you could see when it was closed, when it was ready for re-testing, all of that sort of thing

Critically our interviews suggested that such automated workflow tools were more important in distributed projects than in collocated projects. One suggestion is that the rich-context specific information that these tools bring to bear are more easily available in collocated settings, since it is easy to “walk over to ask a question” or “pull people into a conference room real quick”. This is probably one reason why when managers talk about these tools, they usually add the prefix – global or web-based. These suggest that the real value add from these tools is their availability in all locations that the project functions in, and their platform-independent nature (especially when managers mention web-based tools).

**Why is “Shared” important?:** To truly realize the common ground generation potential of the above tools, the shared repositories, configuration management systems and workflow management tools have to be “truly shared” – i.e., there is one common source of information throughout the project. In a collocated administrative hierarchy, these tools are as important to achieve coordination, especially in large projects and are used to great effect. This is because within a single firm working from a single location it is most likely that the common view of information that shared repositories provides is most complete and accurate, since they are likely to be linked to a single repository.

However, their effectiveness in generating common ground is diminished in most distributed projects. Frequently, different locations, and sometimes different firms working in the same location have their own repositories – i.e., the repository is no longer “shared” by the whole project. Typically, these repositories do not synchronize automatically, but have to be brought to par in what is usually a tedious and error prone manual process. Hence the lack of a fully shared repository among all locations and all vendors in the project results in coordination failures, since common information is not available throughout the project. A manager gave us an example of the coordination problems that occur when many vendors are involved and they do not use a shared configuration management system:

[An] important headache is the mismatch in the technical development tool sets that are not shared by multiple vendors on a single project and the key one that I'll always come back to is configuration management. The more different development groups that you have operating, whether they're within one company or within multiple companies, the greater the risk that you have that your configuration management is not going to be effective. And rarely if ever is there a single consolidated configuration management tool as you were talking about before, that ensures when a particular module is checked up by one person and changes are made, that somebody else isn't making changes to the same code sets that will conflict with those when the application is merged together.

In the above example, every vendor or every location might be using the exact same version of the exact same tool (such as v6.1 of PVCS). However, these two instances of this tool need not be fully synchronized. Not having one integrated tool across all locations and all teams (or tools that could seamlessly talk to each other and keep themselves updated) created several coordination issues. Some examples include giving the same name for different programs, inability to track interdependencies in the code architecture such as a consistent view of the parent and all the child modules, the ability to trace requirements to code, using the wrong version of someone else's code in your fix etc.. These problems multiply in importance and severity especially towards the end of the project when multiple programmers are working on sometimes what they believe to be completely independent change requests.

**Standardization of tools across locations:** We reiterate that the value of the common ground creating tools arises from their being standardized across locations. Our discussions with managers on the above tools usually referred to the need for a "single sand-box environment" as a manager put it. Without such an environment, the effectiveness of the above tools is diminished, since developers in different locations may not observe the same behaviour due to their unique local circumstances. This impairs their ability to understand and anticipate the behaviour of developers in other locations. Put simply, to generate common ground, the tools must be common as well.

However, though provision of standardized tool kits across locations is frequently assumed by developers, though frequently it is not matched by reality. A typical example of a

situation that lacks standardized tool kits is when the onsite and offshore locations use different versions of the same tool. These different versions may cause differences in how the code executes across locations. This difference can be very hard to explain if these differences in environments are not taken into account. One good example from our sample is a team that regularly worked on cutting edge technologies in their project. Developers frequently created prototypes for testing their components and emailed it to their colleagues in remote locations. Many times conversations regarding the different behaviour of these prototypes zeroed in on the slightly different technical environments that were available in each office, which created different behaviour in the code models or prototypes that could not be clearly explained in terms of just the model alone.

Such problems are easier to solve in collocated projects because there is higher awareness of what is happening in the local environment. Such awareness when combined with rich communication easily solves such misunderstandings, while the lack of both awareness and rich communication makes such problems very difficult to solve in distributed projects. Ironically, such problems are more likely to occur in distributed projects than in collocated ones, since each location has a different software upgrade schedule etc. Similarly, such problems are more likely to occur when multiple firms collaborate on a project, since it is easier to achieve and enforce standardization of tool sets and provide training on them etc., within a single firm. In related work, we explore the role of hierarchy in solving such meta-coordination problems as the picking of common standards for ICT tools.

## **DISCUSSION**

In this research, we analysed qualitative data to understand if offshore software services has evolved any features that allows it to compensate for the shortcomings traditionally associated with ICT tools in achieving coordination across distance. We found that the ICT tools that attempt to replicate the richness of face to face communication such as

videoconferencing played only a limited role in remote coordination in offshore software service delivery. Instead, we found that ICT tools are primarily used to achieve tacit coordination – coordination without the need for explicit communication. ICT tools such as shared code repositories, configuration management systems and automated workflow tools generate common ground by enhancing information visibility across locations.

Common ground supports coordination by enabling interdependent individuals to anticipate each others actions and adjust their own behavior accordingly, *without needing to communicate*. While the economic analysis of games typically assumes the presence of common ground (in particular about the rationality of other players and the structure of the game), the shakiness of this assumption is most clearly revealed in empirical studies of games that have multiple Nash equilibria, such as matching games or battle of sexes games (Camerer, 2003). The importance of common ground has been rediscovered in the field of linguistics through the work of Clark (1996) and others, who suggest that every joint act between any two individuals, to be successfully coordinated has to be based on the common ground between them. With every interaction between them, the two individuals generate additional common ground that they could draw upon in subsequent coordination situations.

Clark also suggests that based on previous interactions and based on current task related cues such as what is salient in the environment, interdependent individuals will be able to interpret each others communication accurately in order to achieve coordination. While the economic treatment of common knowledge emphasizes the ability to anticipate actions as the principal benefit, Clark's treatment of common ground emphasizes the ability to interpret communication. Arguably, the two are related – interpretation is the anticipation of meaning in use- but they do differ in outcomes.

The IT tools that we found in use in offshore software services projects enabled coordination across locations by facilitating the generation of common ground. Tools such as

shared repositories, configuration management systems, and automated workflow systems are not tools for communication. Instead, they allow developers in one location to have a view of the work being done in other locations. The essence of this insight may be summarised in the phrase “Who needs to talk if I can see what you see?” A common view across the project generates common ground across the project regarding who is working on what, at what stage they are at, who is interdependent with whom, and in general information regarding the abilities and constraints facing each location. This particular form of common ground, which we call cross-contextual common ground is useful in allowing developers in each location to tailor their activities with respect to members in other locations by anticipating their actions or observing them in real-time with no need for communication. This can be a significant advantage if communication channels are constrained; as is the case for offshore software development. Prior experimental research has found that communicating location specific information to members in other locations is beneficial for task performance in distributed teams (Crampton, 2001; Weisband, 2002). However, much information across locations cannot be verbalized, and communicating even portions of such information could well overwhelm the receivers of such communication. Our findings indicate that IT tools are an excellent means of generating such cross-location awareness as well as indicate which types of information are particularly important.

This ability to coordinate tacitly based on common ground also explains why coordination is difficult when the “configuration management tools are not identical” for instance. These changes provide deviations in the views of the project in each location – deviations that hinder anticipation and observation, and hence adaptation. It is especially a problem when members are not aware of these differences and assume that all project members have identical views to what they themselves see. In such cases, coordination is difficult, since it is difficult to explain where code discrepancies come from. To take a

prosaic example, if you and I are trying to coordinate our movements based on a map, but unknown to both of us, we have different maps we will be unable to coordinate and unable to explain why we are not coordinating. Managers are aware of these difficulties – they take some care to impress upon the developers that the different locations (or vendors) have different tools, and hence they need to take into account these “wrinkles” in common ground.

Our findings also suggest that common ground generating tools and lean media such as telephone and email are complements in achieving coordination, and they together substitute for rich media such as videoconferencing. The ICT tools that enhance common ground, when they do not make communication fully unnecessary, provide the common ground that enables lean media communication using tools such as email and telephone to carry thick enough meanings to be adequate for most coordination purposes. Because such communication builds on pre-existing common ground generated by the other ICT tools, what appears to be “thin” communications to observers actually are very rich in meaning to the participants involved. For instance, it is much easier to and is much more economical in words to give directions to your house to a friend who is familiar with the local area than to one who is new in town. What looked like fairly sparse communication between locations in our projects was able to convey sufficient texture of meaning to get the job done.

In summary, ICT tools mainly enable coordination across remote locations by two means: by generating common ground across locations to make communication unnecessary in many instances, and to allow this common ground to amplify the effects of the communication via relatively poor media such as telephone and email to resolve the remaining coordination problems. In other research we find that the coordination efficacy of cross-contextual common ground generated by ICT tools is enhanced when used in conjunction with other common ground enhancing mechanisms such as using common decision procedures and leveraging prior experience of employees with each other.

Our findings also contribute to a better understanding of the use of technological tools in coordination across locations. These common ground enhancing tools create cross-contextual common ground, both of location specific information as well as information regarding the actions of interdependent members across locations, to achieve coordination without the need to rely on rich media for communication. Prior experimental research has consistently suggested that virtual teams, communicating using ICT technology perform much worse than teams interacting face-to-face (See Kiesler & Cummings, 2002; Kraut et al, 2002 and McLeod, 1996 for reviews). However, offshore software services teams achieve very good performance using ICT tools of low richness. Our findings resolve this contradiction by suggesting that high performance teams using low media can successfully coordinate if cross-contextual common ground is generated by other means. This supports recent research by Crampton (2001) who finds that distributed teams often do not share contextual knowledge about their locations or their activities in their interactions and this frequently leads to low performance and by Mullick et al (2006) who discuss the importance of complementary sets of tools to generate awareness across locations.

Our study also has several implications for practitioners. Remote coordination is not just achieved by facilitating better communication. Given current technological abilities, managers are likely to see greater coordination benefit for their investment if they invest in tools that generate common ground. Hence, instead of investing in videoconferencing, for instance, managers should concentrate more on providing common development environments and tool sets for their developers across several locations. Many managers in our interviews suggested that this was one of their biggest learnings from prior projects. This is ironic, since corporate management in one of the firms that we conducted research in invested in developing a shared conferencing tool across all their locations. However, it is not clear that it has helped much. In most firms, a “global capability” in terms of common

toolsets does not exist, and they do not seem to be aware of the importance of having them for a successful global delivery model. Concentrating on common ground rather than communication is likely to be more useful as firms try to build global delivery models.

This study is subject to a number of limitations. First, we should note the inherent limitations of our method, which lends itself to inductive insights but not generalizations. We chose our industry, offshore software services for reasons of appropriateness rather than representativeness (Miles & Huberman, 1984). Similarly, both the firms that we chose to study and the cases for interviews were selected based on theoretical sampling (Yin, 1992) rather than because they were representative of the industry. While we do present some comparative data, they are not likely to be representative of all projects. Though we looked at a large number of cases within each firm, so that we can be relatively assured about the prevalence of these coordination mechanisms in these firms, they are still small when compared to the volume of projects these firms handle. We cannot assess selection bias and so we cannot make any generalizable claims regarding their performance implications.

Quantitative work on large sample data with longitudinal samples is clearly indicated to understand the prevalence of these tacit coordination mechanisms and their performance implications. The data, especially past projects could also be subject to hindsight bias on the part of our manager informants. However, we tried to mitigate this risk by asking very specific questions relating to specific projects rather than asking the managers to make general statements based on their experience, and also gathered performance data at a later. Finally, since the study is limited to one industry, identifying implicit coordination mechanisms and presence of other common ground enhancers should be studied in other industries with global potential, especially R&D in multi-national corporations.

## **CONCLUSIONS**

This paper explores how coordination occurs in organizations which are characterized by high interdependence between activities as well as constraints on face to face communication because of the distributed nature of activity. Prior research on distributed coordination has examined how ICT tools may help remotely located individuals communicate under such situations and has highlighted the problems with using such tools in-lieu of face-to-face communication. However, our findings emphasize that ICT tools do not enable coordination by communication as much as by creating common ground.

**FIGURE 1 PROJECT SAMPLING STRATEGY**

		Developers belong to SINGLE firm		
		Yes	No	
PROJECT PERSONNEL COLLOCATED IN ONE LOCATION	<b>CELL 1</b> Projects in which all developers (and analysts) work in the same location, and all developers are employed by the vendor.	8 Projects	<b>CELL 2</b> Projects in which all developers (and analysts) work in the same location, but some members are employees of the vendor and others employees of the client or other firms.	18
			10 Projects	
PROJECT PERSONNEL DISTRIBUTED AMONG SEVERAL LOCATIONS	<b>CELL 3</b> Projects conducted by vendor where all developers (and analysts) are employed by the vendor, but they work from several locations (both onshore and offshore)	23 Projects	<b>CELL 4</b> Projects in which developers and analysts work in different locations (project has both onshore and offshore components) and some staff members are employees of vendor, while others are employees of clients or other firms.	42
			19 Projects	
Total		31	29	60

**TABLE 1**  
**OPERATIONALIZING KEY CONSTRUCTS**

<b>Term</b>	<b>Operationalization</b>	<b>Description/Example</b>
Size	Size of the project in terms of budget, FTE, etc., as reported by the manager. We do not judge size – we ask the managers to tell us if it is large, medium or small.	A large development project has about 50 or more developers plus analysts and project management staff. A large maintenance project on the other hand has upward of 15 FTE.
Complexity	Managers report whether the project was either technologically or functionally complex.	Typically, managers reported projects involving novel technologies or functionality they have never worked with before as complex. Also, very large projects (over 100 FTE) were considered complex.
Interdependence between locations	Does the developer in one location write code mindful that this code may inadvertently break the code written by another developer in another location	If the answer to this question is yes, then the project is considered to have high interdependence between locations. It is also considered high interdependence when the analyst in one location and the developer in another have to iterate to create the correct code.
ICT Tools for communication	Information and Communication Technology Tools that enable communication across geographic distance	Telephone, mobile phones, Instant Messenger, Email, NET Meeting, Live Meeting, Electronic Whiteboards, computer conferencing, Video conferencing, Web Cam technology etc.
Project performance	Defined in terms of how well did the project meet pre-defined objectives	Adherence to its original schedule and budget, meeting service level agreements, as well as how well the developed software captured the functionality required measured by client satisfaction with the project.

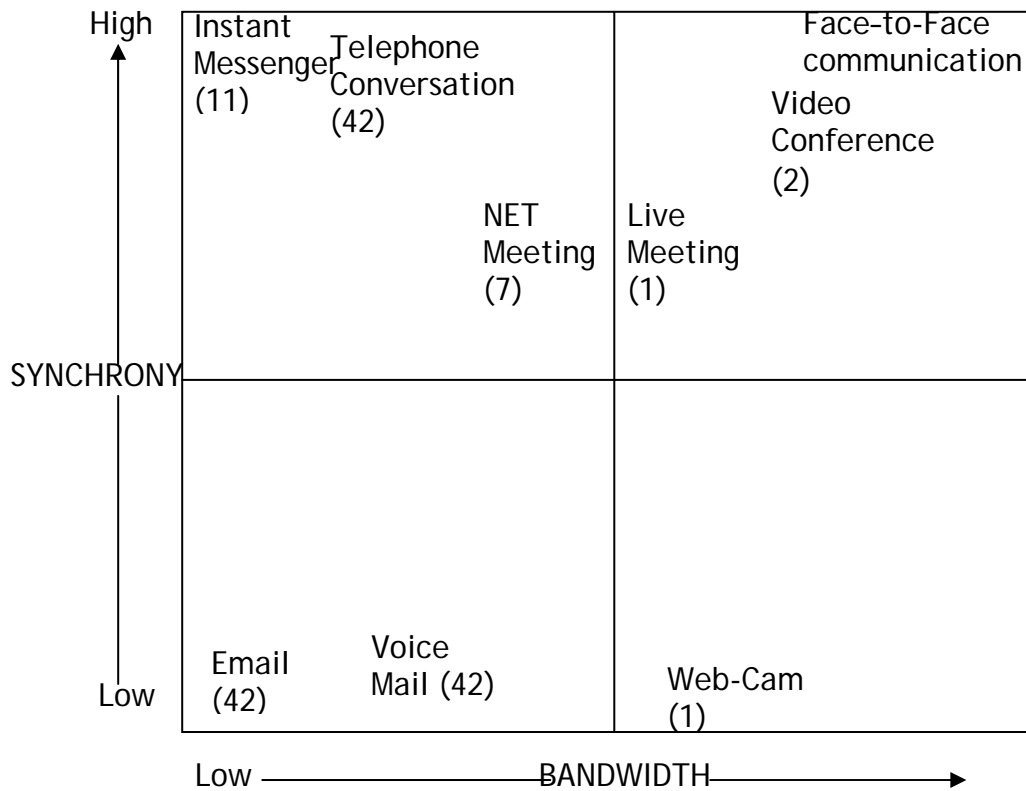
**TABLE 2**  
PROJECT CHARACTERISTICS

		PROJECT SIZE			Totals	
		SMALL	MEDIUM	LARGE		
LOCATION MODEL	MULTIPLE	High Complexity	0	7	26	33
		Low Complexity	3	6	0	9
	SINGLE	High Complexity	2	4	4	10
		Low Complexity	2	3	3	8
Totals		7	20	33	60	

**TABLE 3**  
PROJECT PERFORMANCE

		PROJECT SIZE			Totals	
		SMALL	MEDIUM	LARGE		
LOCATION MODEL	MULTIPLE	Good Performance	3	12	20	35
		Poor Performance	0	1	6	7
	SINGLE	Good Performance	4	6	5	15
		Poor Performance	0	1	2	3
Totals		7	20	33	60	

**FIGURE 2**  
**DISTRIBUTION OF ICT TOOLS IN THE MEDIA RICHNESS SPACE**



Synchrony: The ability to interact in real time and receive immediate feedback  
 Bandwidth: The ability to receive visual cues

## REFERENCES

- Adler, P.S., 2005 "Beyond hacker idiocy: A new community in software development," in C. Heckscher and P.S. Adler, eds. *The Firm as a Collaborative Community: Reconstructing Trust in the Knowledge Economy*, Oxford, UK: Oxford University Press.
- Armstrong DJ and Cole P 2002 Managing distances and differences in geographically distributed work groups pp167-186 In Hinds and Kiesler (Eds.) *Distributed Work*, Cambridge MIT Press.
- Athreye, S. 2005 The Indian Software Industry and its evolving service capability, *Industrial and Corporate Change* 14(3): 393-418.
- Aumann R and Brandenburger A 1995 Epistemic conditions for nash equilibrium *Econometrica* 63(5): 1161-1180.
- Baldwin, C Y and Clark, K. 2001, *Design Rules: The Power of Modularity*, Harvard Business School Press, Harvard University, MA
- Camerer C 2003 *Behavioural Game Theory*, Princeton NJ: Princeton University Press
- Cataldo, M., Wagstrom, P., Herbsleb, JD., Carley, K. 2006 Identification of coordination requirements: Implications for the design of collaboration and awareness tools. *Proceedings of Computer-Supported Cooperative Work* 2006.(forthcoming)
- Crampton, C. (2001). The mutual knowledge problem and its consequences for dispersed collaboration. *Organization Science*, 12(3), 346-371.
- Clark, H. 1996 *Using Language*, Cambridge, UK: Cambridge University Press.
- Clark H and Brennan SE 1990 Grounding in Communication. In LB Resnick, RM Levine and SD Teasley (Eds) *Perspectives on Socially Shared Cognition* (127-149) Washington DC: American Psychological Association.
- Daft, R. and Lengel 1984 Information richness: a new approach to manage information processing and organization design in Staw and Cummings (eds) *Research on Organizational Behavior*, Greenwich, CT: JAI Press.: 191-233
- Daft, R. and Lengel 1986. Organizational Information Requirements, Media Richness & Structural Design, *Management Science* 32(5): 554-571
- De Meyer A. 1991 Tech talk: How managers are stimulating global R&D communication. *Sloan Management Review* 37: 49-58.
- Doherty-Sneddon G Anderson A O'Malley C Langton S Garrod S and Bruce V 1997 Face-to-face and video mediated communication: A comparison of dialog structure and task performance. *Journal of Experimental Psychology* 3:105-125.
- Eisenhardt, K. 1989 Building theories from case study research, *Academy of Management Review*, 14:488-511.
- Ethiraj, S Kale, P Krishnan, M and Singh J. 2005 Where do capabilities come from and how do they matter? A study in the software services industry, *Strategic Management Journal* 26(1): 25-45.
- Ethiraj, S. and Levinthal, D. 2004 Modularity and innovation in complex systems, *Management Science*, 50(2): 159-163.
- Evans, P. and Wurster, T. 1999 *Blown to Bits: How the New Economics of Information Transforms Strategy*, Harvard Business School Press
- Fish RS Kraut RE and Chalfonte BL 1990 The VideoWindow system in informal communication, In *Proceedings, Computer Supported Cooperative Work* (1-12). New York: ACM Press.
- Fish, R. S., Kraut, R. E., Root, R. W., Rice, R. 1993. Video as a technology for informal communication. *Communications of the ACM*, 36(1), 48-61.
- Friedman, Tom 2005 *The World is Flat* Farrar, Strauss and Giroux
- Galbraith, J. 1977 *Organization Design*, Addison-Wesley Publishing Company.

- Glaser, B. and Strauss, A. 1967 *The discovery of grounded theory: Strategies for qualitative research*, Aldine de Gruyter, New York.
- Grandori A 1997 Governance Structures, Coordination Mechanisms and Cognitive Models. *Journal of Management & Governance*, 1(1): p29-47
- Hackman JR and Oldham GR 1980 *Work Redesign* Reading, MA: Addison-Wesley.
- Hagel, H. and Singer, M. 1999 Unbundling the corporation, *Harvard Business Review* 77(2): 133-141
- Heath C and Luff P 1991 Disembodies contact: Communciation through video in amulti-media environment. In *Proceedings of the CHI 91 Conference on computer human interaction* p:99-103. New York: ACM Press.
- Heath, C. and Staudenmayer, N. 2000 Coordination Neglect: How Lay theories of organizing complicate coordination in organizations, In R.I. Sutton and B. Staw (eds.), *Research in Organization Behaviour*, vol 22, Greenwich, CT: JAI Press pp153-193.
- Herbsleb, J., Paulish, D.J., Bass, M. 2005 Global Software Development at Siemens: Experience from Nine Projects. *International Conference on Software Engineering (ICSE)*, pp. 524 - 533, St. Louis, MO, May 15-21.
- Herbsleb, J.D. & Mockus, A. 2003. Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering. In proceedings, *European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 112-121, Helsinki, Finland, September 1-5, 2003.
- Herbsleb, J.D. & Mockus, A. 2003 An Empirical Study of Speed and Communication in Globally-Distributed Software Development 2003. *IEEE Transactions on Software Engineering*, 29, 3, pp. 1-14.
- Hinds, P.J., & Kiesler, S. (Eds.). (2002). *Distributed work*. Cambridge: MIT Press.
- Kiesler, S. and Cummings, J. 2002 What do we know about proximity and distance in work groups? A legacy of Research. In Hinds and Kiesler (Eds.) *Distributed Work*, Cambridge MIT Press.
- Kiesler, S. and Sproull, L 1992 Group Decision Making and Communication technology, *Organizational Behavior and Human Decision Processes*, 52:96-123.
- Krauss RM and Bricker PD 1966 Effects of transmission delay and access delay on the efficiency of verbal communication *Journal of Acoustical Society* 41: 286-292.
- Krauss, R.M. and Fussel, S.R. 1991 Perspective taking in communication: Representations of others' knowledge in reference, *Social Cognition*, 9:2-24.
- Kraut RE Galegher J and Egidio C 1988 Relationships and tasks in scientific research collaborations. *Human Computer Interaction* 3:31-58.
- Kraut, R. Fussell, S. Brennan, S. Siegel, J. 2002 Understanding effects of proximity on collaboration: Implications for technologies to support remote collaborative work. In Hinds and Kiesler (Eds.) *Distributed Work*, Cambridge MIT Press.
- Kraut, R. and Streeter, L. 1995. Coordination in large scale software development. *Communications of the ACM*, 38(3), 69-81
- Lawrence, P. and Lorsch, J. 1967 *Organization and environment: Managing differentiation and integration*. Boston, MA: Harvard University Press.
- March JG, Simon HA. 1958. *Organizations*. Wiley Press.: New York
- McGuire T Kiesler S Siegel J 1987 Group and compute mediated discussion effects in risk decision making *Journal of Personality and Social Psychology* 52:917-930
- McLeod, P 1996 New Communication technologies for group decision making: toward an integrative framework, in Hirokawa and Poole (eds.) *Communication in Group Decision Making*, 2<sup>nd</sup> Edition

- Mockus, A., Fielding, R., & Herbsleb, J.D. Two Case Studies of Open Source Software Development: Apache and Mozilla (2002). *ACM Transactions on Software Engineering and Methodology*, 11(3): 309-346.
- Moon J. and Sproull, L. 2002 Essence of distributed work: The case of the Linux kernel. In Hinds and Kiesler (Eds.) *Distributed Work*, Cambridge MIT Press.
- Miles, M. and Huberman, M. 1994 *Qualitative data analysis*. Thousand Oaks, CA: Sage.
- Mullick, N., Bass, M., Houda, Z., Sangwan, R., Paulish, D., Cataldo, M., Herbsleb, J., Bass, L. 2006 Siemens Global Studio Project: Experiences adopting an integrated GSD infrastructure. *Proceedings of the International Conference on Global Software Engineering*. (forthcoming)
- Nadler, D. and Tushman, M. 1998 Competing by Design. *Executive Excellence*, 15(8):12
- O'Connell B Whittaker S and Wilbur S 1993 Conversations over videoconferences: An evaluation of the spoken aspects of video mediated interaction. *Human Computer Interaction* 8:389-428
- Olson, G. and Olson, J. 2000 Distance Matters *Human Computer Interaction* 7 347-374.
- Olson JS Teasley S Covi L Olson G The (currently) unique advantages of collocated work pp 113-136. In Hinds and Kiesler (Eds.) *Distributed Work*, Cambridge MIT Press.
- Orlikowski, W. 2002. Knowing in Practice: Enacting a Collective Capability in Distributed organizing, *Organization Science* 13(3):249-273
- Parnas D.L., Clements, P.C., and Weiss, D.M. 1981 The modular structure of complex systems, *IEEE Transactions on Software Engineering*, SE-11, No. 3, pp 259-266.
- Puranam, P. and Gulati, R. 2005 Bringing 'Coordination' back in to the study of inter-organizational collaboration, *London Business School Working Paper*. 2005: Jan. 1-33.
- Rice, R.E., 1993 Media Appropriateness: Using social presence theory to compare traditional and new organizational media, *Human Communication Research*, 19: 451-484.
- Rivkin J and Siggelkow N 2003 Balancing Search and Stability: Interdependencies Among Elements of Organizational Design. *Management Science*, 49(3):290-311
- Schelling TC 1960 *The Strategy of Conflict* Cambridge, MA: Harvard University Press.
- Siegel J. Dubrovsky V., Kiesler, S. And McGuire T. 1986 Group processes in computer-mediated communications. *Organizational Behavior and Human Decision Processes* 37: 157-187.
- Simon, H.A. 1962 *The Sciences of the Artificial*, Cambridge, MA: MIT Press.
- Short J. Williams E. and Christie, B. 1976 *The social psychology of telecommunications*. New York: Wiley.
- Strauss, A., and Corbin, J. 1997. *Basics of qualitative research: Grounded theory procedures and techniques*. Newbury Park, CA: Sage.
- Thompson JD. 1967. *Organizations in Action*. McGraw Hill.: New York
- Tushman, M. and Nadler, D. 1978 Information Processing as an integrating concept in organization design, *Academy of Management Review* 3:613-624.
- Wageman, 2003 Virtual processes: Implications for coaching the virtual team, in Peterson and Mannix (Eds.) *Leading and managing people in dynamic organizations*. Mahwah, NJ: Erlbaum
- Walther, J. 2002 Time effects of computer-mediated groups: Past, present and future pp235-258. In Hinds and Kiesler (Eds.) *Distributed Work*, Cambridge MIT Press.
- Weber, R 2004 Some Implications of the Year-2000 Era, Dot-com Era, and Offshoring for Information Systems. *MIS Quarterly*, 28(2): iii-xi
- Weisband, S. 2002 Maintaining awareness in distributed team collaborations: Implications for leadership and performance pp 311-334. In Hinds and Kiesler (Eds.) *Distributed Work*, Cambridge MIT Press.

Whittaker S 1995 Rethinking video as a technology for interpersonal communication  
*International Journal of Human Computer Studies* 42:501-529  
Yin, R. 1994 *Case Study research: Design and Methods*. Beverly Hills, CA: Sage.

---

<sup>i</sup> OECD Working party on the Information economy, April 2005 – reported in the PriceWaterhouseCoopers and Economist Intelligence Unit joint report on offshoring the financial services industry, September 2005.

<sup>ii</sup> “A world of work: A survey of outsourcing” The Economist, Nov 13, 2004, p4.

<sup>iii</sup> NASSCOM Strategic Review, 2006

<sup>iv</sup> As evidence of the relative neglect of the role of ICT in geographic dispersion of activities rather than across firm boundaries, consider the following: We searched three premier IS journals (MIS Quarterly, Journal of Management Information Systems and Information Systems Research) in Business Source Premier for articles published between 1990 and the latest articles in the database (typically, Dec 2005) with the phrase “outsourcing” or “vendor” and got 51 articles. To identify articles on offshoring, we used the much more complicated and exhaustive search phrase: “remote coordination” OR “distributed coordination” OR “distributed work” OR “remote work” OR “remote groups or teams” OR “distributed groups or teams” OR “virtual groups or teams” OR “distributed software” OR “remote software” OR “distributed application” OR “offshor\*” OR (geographic\* AND (dispersed or coordination)) and got a mere 14 articles, of which only 2 were field based empirical work.

<sup>v</sup> Project performance was measured as the achievement of stated objectives. The measures included were adherence to budget, adherence to the original schedule, meeting pre-defined service level agreements, and a subjective measure of the quality of the software in terms of client satisfaction and vendor satisfaction with the project. It is well known that distributed projects usually take longer to complete than collocated projects (Herbsleb et al, 2005; Herbsleb and Mockus, 2003) due to coordination overheads, and in our sample managers did consciously choose longer schedules for distributed work. However, we are interested in successful coordination as in meeting pre-defined objectives.