COORDINATING DISTRIBUTED SOFTWARE DEVELOPMENT: A RESOURCE RELATIONSHIPS PERSPECTIVE ON ANALYZING THE SPATIAL EFFECTS

Completed Research Paper

Gamel O. Wiredu
Ghana Institute of Management and Public Administration (GIMPA)
PO Box AH50, Achimota – Accra, Ghana
gwiredu@gimpa.edu.gh

Abstract

As more software development organizations are increasingly distributing their operations spatially, information systems development researchers are taking perspectives such as transactions costs and resource dependency to explain the effects of spatial distribution on coordination. This paper argues that these perspectives are limited because they do not address all the key relationships between software development resources in a unified and systemic manner. The interactions between people, information, and technology, which are the key software development resources, are characterized by four key relationships – interdependencies, uncertainties, conflicts, and technology representations. Based on the premise that spatial distribution of the resources exposes their relationships to direct environmental stimuli, the paper proposes a complementary resource relationships perspective to explain the spatial effects on the dynamics of coordination. An empirical example of software development distributed between sites in USA and Republic of Ireland is used to illustrate these issues. Analyzing the effects of spatial distribution in terms of resource relationships leads to a more systemic, in-depth and phenomenological understanding of the dynamics of coordination. Implications of this perspective for the analysis of coordination in distributed software development are discussed.

Keywords: Resource relationships, coordination, interdependencies, uncertainties, conflicts, technology representations, distributed software development
Introduction

Spatially distributed software development (DSD) represents a considerable transformation of software development, exemplified by outsourcing, ‘offshoring’ and ‘nearshoring’ of work across national borders (Carmel and Abbott 2007; Carmel and Tjia 2005; Sahay et al. 2003). Spatial distribution, which is the contemporary and independent factor, challenges many existing theories of software development coordination. This factor exposes development resources – people, technology and information – to direct environmental stimuli, and represents an important structural change in software development operations. The factor increases the complexity of the interdependencies between the resources (Espinosa et al. 2007; Herbsleb and Grint 1999) as well as the related uncertainties and conflicts. Because coordination is the management of interdependencies (Malone and Crowston 1994), its practice and theory in the face of spatial distribution is a significant challenge. Kraut and Streeter (1995), for example, indicate that a major contributor to software development crisis is the problem of coordinating activities.

The coordination challenges of DSD have recently attracted attention in information systems development research. Existing research explains the effects of spatial distribution on software development coordination in terms of challenges such as communication delays (Herbsleb and Mockus 2003), communication breakdowns (Herbsleb and Grint 1999), and knowledge sharing or transfer problems (Espinosa et al. 2007; Kotlarzky and Oshri 2005; Oshri et al. 2008). These challenges are usually compared with benefits to be obtained from 24-hour or follow-the-sun development (e.g. Grint et al. 1999; Trienen and Miller-Frost 2006), closer-to-market development (e.g. Casey and Richardson 2004), and access to a global pool of technical and experienced developers (e.g. Ebert and De Neve 2001). Interestingly, follow-the-sun, closer-to-market, and communication speed benefits as well as communication and knowledge problems are all explanations of spatial distribution effects in terms of transactions costs or economic efficiency (Williamson 1975). On the other hand, benefits of access to a global pool of technical and experienced developers are explanations of the effects in terms of resource dependency (Pfeffer and Salancik 1978). However, transactions costs and resource dependency explanations of spatial distribution effects are limited because of the following reason.

They lead researchers to focus differently and narrowly on interdependencies, uncertainties, or conflicts. But a focus on individual relationships addresses only aspects of the spatial effects on coordination, and renders the explanations incomplete. “Interdependence characterizes the relationship between the agents creating an outcome” (Pfeffer and Salancik 1978, p.40); but so also do uncertainty and conflict (Jehn 1997; Thompson 2003). They all characterize relationships between software development resources – developers, technology and information. Because of this narrow focus, existing explanations do not help to explain sufficiently resource interdependencies as a function of other relationships that are also affected significantly by spatial distribution. Yet, coordination problems of DSD may arise from all the relationships. Besides, they do not capture phenomena that lie beyond dependency and efficiency. As a result of these limitations, information systems development literature lacks an elaborate and holistic conceptualization of how spatial distribution of software development resources affects their coordination.

The understanding of spatial distribution effects on resource relationships requires a framework of resource relationships. This paper, therefore, draws upon the theory of coupling (Glassman 1973; Orton and Weick 1990) to develop a resource relationships perspective that proffers unified and systemic explanations of spatial distribution effects on coordination. Coupling theory is useful for the study of organization structure as it facilitates explanations of how the resources of a system directly or indirectly affect each other in their interrelations. This paper’s emphasis on relationships, beyond dependency and efficiency, contributes a distinctive yet complementary perspective by explaining the effects of spatial distribution on all key resource relationships in a single effort. The perspective also suggests a phenomenological orientation towards the study of the interrelations between people, information, and technology in DSD. Thus, beyond dependency and efficiency, the paper foments analysis of phenomena such as agility, drift, and adaptation to changing requirements. This is illustrated by an empirical example of software development distributed between three sites in USA and one site in the Republic of Ireland. The paper is, therefore, informed by Thompson’s (1956, p.104) argument that “obtaining greater precision in the statement of relationships among phenomena under stated conditions” is a very important aspect of science.

In developing this perspective, software development is being perceived as a complex task because of two key reasons. First, it is a task characterised by exceptional problems that are difficult to analyze (Perrow 1967; Van de Ven and Delbecq 1974). Second, it is a task whose information requirements are characterized by high degrees of uncertainty (Brooks 1995; Kraut and Streeter 1995). These characteristics are typical of research and development.
(R&D) work (see, for example, Allen and Cohen 1969; Tushman 1978; Tushman 1979), and they contrast sharply with routine organizational tasks (Daft and Lengel 1986; Perrow 1967). In respect of this, the resource relationships perspective is also important because it facilitates a modelling of the dynamics of coordination of DSD in terms of how the resource relationships in focus further relates with the complexity of software development.

The paper’s contributions are important both theoretically and practically because DSD is fast becoming the new order in contemporary software development. These days, many North American and Western European organizations are outsourcing some of their operations to countries such as India and China to take advantage of cheaper skilled labour. Recently, skilled labour has become very expensive in the West, and has induced organizations to relocate even core operations to the Far East. Such a trend seems to depend on advancements in information and communication technologies that were predicted by M. Castells in “The Rise of the Network Society” (Castells 1996). Kallinikos (2006, p.9), for example, sees it as “the quest for alternative economic and organizational practices combine[d] with the impressive instrumental involvement of information and communication technologies.” Castells dreamed that such technologies would overcome distance and time barriers, and companies would benefit from 24-hour continuous work progress in different offices around the globe. However, while this may achieve direct labour cost savings, the same cannot be said of the costs of overcoming the coordination challenges posed by distance, environmental and socio-cultural constraints. Theoretically, the current level of systems development research on coordination in the domain of DSD seems to lag behind its practice. The development or extension of knowledge about practice, and the application of that knowledge to the practice is important (Van de Ven 1989). Thus, the resource relationships perspective being proposed in this paper contributes greater understanding of this concept, and stimulates more research attention towards it.

The next section presents a short review of existing coordination perspectives and their synthesis which show that they are closely interrelated. The following section explains how and why spatial distribution opens up the software resources in the organization’s operating core to environmental stimuli; and, subsequently, undermines the relationships between these resources. After this, the theory of coupling is used to conceptualize the coordination challenges engendered by the opening and undermining effects, as well as the ways by which organizations should address them, thereby consummating the resource relationships perspective. Then the implications of the perspective for the analysis of coordination in DSD are discussed. The paper closes with concluding remarks.

A Synthesis of Existing Coordination Constructs

Coordination is not a new concept in information systems and organizational research. It was the underpinning philosophy of Taylorism, and has remained the preoccupation of organizational researchers who have grappled with it over the years (e.g. Malone and Crowston 1990; Malone and Crowston 1994; March and Simon 1993; McCann and Ferry 1979; Mintzberg 1983; Montoya-Weiss et al. 2001; Quinn and Dutton 2005; Schmidt and Simone 1996; Thompson 2003; Van de Ven et al. 1976). It is not my aim to do a comprehensive review of coordination in this paper; and therefore I will only summarize the predominant constructs of the concept that are useful as bases for the subsequent deliberations.

While many researchers (e.g. Malone and Crowston 1994; March and Simon 1993; Thompson 2003; Van de Ven et al. 1976; Weick 1979) theorize coordination in terms of managing interdependencies, others (e.g. Daft and Lengel 1986; Milliken 1987) explain it in terms of managing uncertainties and equivocalities. Some (e.g. Montoya-Weiss et al. 2001; Schmidt and Kochan 1972; Victor and Blackburn 1987) have theorized it in terms of interpersonal and interunit conflict management; while researchers in Computer-Supported Cooperative Work (CSCW) (e.g. Carstensen and Sørensen 1996; Schmidt and Simone 1996) explain it in terms of managing technology representations. ‘Technology representations’ refer to the functional role of technology in coordinating organizational operations. “Representations are social facts” (Rabinow 1998), and technology representations connote the socio-technical role either assigned to or obtainable from technology. In an optimal representation within the context of coordination, the technology functions optimally in managing conflicts and uncertainties, and in addressing interdependence bottlenecks.

Interdependencies, uncertainties, conflicts, and technological representations signify relationships between software development resources – people, information, and technology. The coordination literature shows that theoreticians focus more attention on interdependent relations than on uncertainties, conflicts or technology representations. However, the arguments of this paper are founded on equal attention to all four relationships. There are two main reasons that justify this foundation.
First, significant references to uncertainties, conflicts and uncertainties are made implicitly and explicitly in most discussions of interdependencies in organizational research literature. For example, Daft and Lengel (1986) argue that “[i]nterdependence increases uncertainty because action by one department can unexpectedly force adaptation by other departments in the production chain” (p.565); Thompson (2003, p.138) argues that the “[p]otential for conflict … increases with interdependence;” and Schmidt and Simone’s (1996) discussion of coordination mechanisms talk about artifacts and protocols as representations of technology that are essential for supporting interdependent work. Furthermore, Daft and Lengel’s research confirms that uncertainties and interdependencies are closely interrelated; Jehn’s (1995; 1997) discussion of conflict types relate implicitly to uncertainties and interdependencies; and Tushman and Nadler’s (1978) and Galbraith’s (1977) discussions about information processing relate technology with uncertainties and interdependencies closely. To wit, increased uncertainties can increase conflicts and further undermine interdependencies. However, conflicts can worsen uncertainties and undermine interdependencies; or the perception of goal incompatibility coupled with perceived opportunity for interference by interdependent units can lead to overt or covert conflicts. Poorly functioning technologies can also increase uncertainties, conflicts and interdependence problems; while such problems can also lead to frustrations with the use of technologies.

Second, the numerous documented challenges that confront distributed software development suggest that conflicts, uncertainties, and technology representations cannot be relegated to secondary statuses in coordination analysis. For instance, a lot has been written about conflicts and uncertainties between distributed teams working interdependently (e.g. Cramton and Hinds 2005; Hinds and Bailey 2003; Jarvenpaa and Leidner 1999; Mannix et al. 2002; Sahay et al. 2003). Furthermore, given the essential role of information and communication technology in enabling DSD, technology representations that can enhance interdependencies, and reduce conflicts and uncertainties are considered primary in coordination analysis.

These two reasons show that each of these four relationships significantly bears on the other, and the degeneration of any can cause coordination problems, especially in DSD. The degeneration, in turn, is caused by spatial distribution of the software development resources that lie beneath the relationships.

The Effects of Spatial Distribution on Software Development Resources

Organizations are not the same. Scott’s (2003) review of organization genres - closed-rational, closed-natural, open-rational and open-natural systems - illustrates the diversity in organizations. Before Scott, however, Thompson (2003) had argued against such categorization of organizations because it stifles thinking about the concurrency of openness and closeness in organizations. He therefore conceptualized “complex organizations” as determinate and closed systems with an orientation to reduce uncertainty and, at the same time, as indeterminate and open to inevitable uncertainties from the environment. Thus, a complex organization is “an open system subject to criteria of rationality” (2003: 11). In this paper, the analysis of how spatial distribution of software development operations affects coordination is also founded on the conception of a complex organization. Drawing on Parsons (1960), Thompson argued that a complex organization is less rational and more open to environmental influences at its institutional level; it is more closed and rational at its technical core; and at its mediating managerial level, it is characterized by the interplay of openness and closeness (see Figure 1). Thus, although my conceptualization focuses on the dynamics of coordination of distributed software development operations, it incorporates analysis of the effects of the managerial, institutional and environmental influences on the technical core.

When collocated software development resources (site A) are distributed (resulting in sites A and B), parts of the organization’s operating core are opened to the environment (see Figure 1). By virtue of spatial distribution, site A’s operations may still be more protected from direct environmental influences by its managerial and institutional levels than site B’s. However, so long as operations in sites A and B are interdependent, site A is essentially open to the environment since environmental stimuli affecting site B would undoubtedly trickle down to affect site A. Distributing software development operations will undermine the managerial and institutional buffers that the resources for those operations enjoyed when they were collocated. Therefore, the organizing logic of its operations changes from closed to partly closed and partly open, and will affect the DSD resources.
Figure 1: The significance of ‘distribution’ in distributed software development [The arrows indicate information flows]

Software Development Resources – People, Information and Technology

The people in a DSD activity include all those individuals such as programmers, testers, project managers and customers whose concerted efforts require coordination. The processes they engage in, the technologies they adopt and use in these processes, as well as the range of information they generate, process and transmit with these technologies must be coordinated. The software development process is decisive for success because it “is a dialogue in which the knowledge that must become the software is brought together and embodied in the software. The process involves interaction between users and designers, between users and evolving tools, and between designers and evolving tools” (Baetjer Jr 1998, p.85). Processes are modes of operating (Mathiassen and Stage 1992) that include all the tasks such as modelling, programming and testing; all modes of interactions. They also include information generation, processing and transmission tasks (see Humphrey 1989). In a distributed development environment, people and processes may be affected by the socio-cultural characteristics of the particular sites in which they operate. These may condition their belief systems, reflect in their “frames of reference” (Gioia 1986), reflect in their attitudes, and subsequently engender conflicts and erratic interdependent relationships.

The resource essence of information in software development lies in its role as the lifeblood that circulates to ensure the functional relationship between people and technology. It is the source of knowledge, and the resource upon which relationships such as interdependencies, uncertainties, and technology representations gain their meaning. The information resource is understood in terms of form, modes of capture, knowledge repositories and sharing modes, processing and transmission. The accuracy, timeliness, reliability, completeness, and sufficiency of information may be affected by spatial distribution. Besides, distributed developers, by virtue of their different frames of reference, may interpret information differently, leading to conflicts.

The technology resource captures all forms of technology artefacts that are deployed in support of the interactions between people and information. A distinction is made between technologies which are used to develop software, and technologies which are adopted and deployed to coordinate the development. Development technologies such as languages and platforms, bug tracking systems and knowledge repositories can be affected by distance because people hired to work in various sites may have different preferences for some technologies. Likewise, technologies that facilitate coordination are information generation, processing, and interaction systems used for overcoming space and time barriers. They can limit fact-to-face, spontaneous and informal interactions between developers.

In sum, spatial distribution effects on these resources are understood in terms of the coordination challenges brought by all these relationships between them – interdependencies, uncertainties, conflicts, and technology representations. The exposition of these resources to environmental stimuli, the ensuing coordination challenges, and the ways by which organizations should respond to them are next conceptualized through the theory of coupling.
A Resource Relationships Perspective on Analyzing the Effects of Spatial Distribution on Coordination

The degree of coupling between resources of a system or between two systems is understood as how those resources or systems affect each other in terms of direct or indirect causal impact (see Glassman 1973; Orton and Weick 1990; Weick 1976). Thus when effects are direct, then those resources or systems are deemed to exhibit tight coupling; and vice versa. Due to the closed organizing logic of collocated software development operations, the coupling between people, technology, and information is understood in terms of the direct causal impact and continuous connection between them. Thus, the operations will exhibit a tightly coupled system, while their coupling with the environment will be loose. This logic presumes minimal environmental interference in the interrelations between the resources. But when software development is spatially distributed, the coupling between these resources and their environment gets tight because the environmental influences become more direct than the scenario in collocated organizing. These direct influences bear on the distributed resources by mediating the erstwhile direct relationships between them. In short, the relationships receive and respond to stimuli in the organization’s environment. Moreover, because of the organizational natural predilection for optimal coordination, any stimulus that orients the system towards a suboptimal state must be processed by managing the relationships to reverse it.

The coordination implications of the loose coupling between the distributed resources on the one hand, and their tight coupling with the environment on the other, are now conceptualised. This section examines how these coexisting and contrasting couplings affect resource relationships and their management in DSD operations. This examination is done in respect of organisation’s orientation towards two1 desired organizational outcomes of loose coupling – adaptability and persistence (Orton and Weick 1990).

Adaptability and Persistence in Managing Resource Relationships

Firstly, loose coupling in DSD operations will be managed to strive ultimately towards persistence in its environment. This will reflect in adaptation to the environment, and stability and continuity of operations via the neutralization of change. Secondly, within the purview of striving for persistence, coordination strategy will aim, in short-term cycles, for adaptability to environmental uncertainties. This will reflect in its ability to assimilate and accommodate diverse and direct environmental changes (Weick 1979). Therefore, adaptability is understood as continuous development of diverse capacities for responding to environmental stimuli; while persistence is understood as eventual minimization of the diversity of direct environmental stimuli that will reduce the resource relationships’ vulnerability to them.

Persistence is more of a medium- to long-term aim compared with adaptability which is short-term. Software organisations will orient DSD operations towards closure of distributed operations to direct environmental influences in the long run. Therefore, its short-term cycles of adaptability will be designed to fulfil the long-term aim of persistence. Since adaptation is likely to preclude adaptability (Weick 1976; Weick 1979), software organisations will institute measures to ensure that adaptability is not sacrificed for adaptation in the long run. This is true because even when the distributed operating core has adapted to its environment, there is always the likelihood of changes in the institutional and material-resource environments that will necessitate continuous cycles of re-adaptation. Coupling is now used to explain the effects of spatial distribution on resource relationships in software development.

Interdependencies

In the closed logic, because operations are buffered from direct environmental effects to ensure a high degree of operational determinacy, interdependent relations between the resources are more direct. However, when parts of such interdependent relations are spatially distributed to open them to the environment, they bear inter-site and environmental effects. Inter-site interdependencies are those between developers operating in different sites. Environmental interdependencies are those that are engendered by fluctuations in input sources and output markets.

1 The other three outcomes – buffering, satisfaction, and effectiveness – discussed by Orton and Weick are excluded because first, buffering is synonymous to persistence in the context of software development; second, satisfaction pertains to employees and, therefore, to resources but not to their relationships; and third, effectiveness is a natural corollary of all the other four outcomes.
Once some software development operations are moved to a new location, the internal interdependent relationships between their resources will be mediated by direct environmental dependencies, some of which may be adverse. This means that interdependent relations between the resources will change from direct to indirect, and those between the resources and the environment will change from direct to direct.

The achievement of adaptability and persistence requires continuous management of environmental and inter-site interdependencies in tandem. To manage inter-site interdependencies, related conflicts and uncertainties are reduced by collocating components or modules of the software. This, in effect, reduces task interdependencies between developers in different sites. At the same time, the frequency of communications between sites may be increased in order to increase shared knowledge and reduce conflicts. This is normally done through teleconferencing where developers are kept continuously aware of what is happening in other sites so that they are always on the same page (Levesque et al. 2001). These measures are implemented continuously to ensure that information and task exchanges between distributed locations remain functional. They are also essential to ensuring persistence in the face of the vulnerability of inter-site interdependencies to the environment.

Environmental dependencies may be managed through continuous monitoring of material input sources, labour sources and markets obtained from the environment, and continuous seeking of new ones with the aim of dispersing them. Dispersal of environmental dependencies is a means of reducing the power of the environment over core operations (Dill 1958; Thompson 2003). It is also a means of ensuring that distributed core operations are more certain. However, in the instance when environmental dependencies are concentrated inevitably, managers will follow Thompson’s suggestion to contract, coopt or coalesce input sources and output markets to reduce the vulnerability of operations to material-resource environmental fluctuations. The dispersal of environmental dependencies implies the need for the software organization’s capacity to tune and retune continuously the operations to match the dispersal. While continuous tuning and retuning reflects the capacity for adaptability, optimal dispersal of environmental dependencies reflects the capacity for persistence. Managers will exploit the loose coupling of software development resources to facilitate the development of this capacity.

**Uncertainties**

The closed logic of collocated operations also implies low degrees of uncertainty in software development resource relationships. It implies, for example, the software organization’s achievement of high security for inputs, throughputs and outputs. However, when operations are distributed, site B’s resources would receive direct stimuli from its institutional and material-resource (or task) environments (Dill 1958; Meyer and Rowan 1977; Scott 2003). Institutional environments such as national legal frameworks and societal norms of site B’s region, as well as material-resource environments such as input sources and output markets are sources of operational uncertainties. Institutional environments coerce organizations to remodel their structures mostly to gain and maintain legitimacy and support. Thus, material-resource or task environments have significant bearings on the certainty of the relations between distributed software development resources. In this regard, when an organization has little influence or power over such environments, and when the stimuli thereof are borne directly by the resources, then high uncertainties between them are likely. Whereas in collocated software development, resource relationships are buffered from fluctuations in the institutional and material-resource environments, DSD is more vulnerable to fluctuations because of their openness to the environment.

To ensure adaptability of core operations, managers adopt multiple communication modes, deploy quality communications technologies and institute processes to operationalize those modes. For multiple communication modes such as teleconferencing, e-mailing, instant messaging and telephone calling, each one has different characteristics yet those characteristics complement each other in terms of reducing information uncertainties. Thus teleconferencing is normally synchronous, broadcast or many-to-many, and unobtrusive, with ephemeral information; e-mailing is normally asynchronous, one-to-one or broadcast, and obtrusive or unobtrusive, with persistent information; telephone calling is normally synchronous, one-to-one, and obtrusive, with ephemeral information; and instant messaging is normally synchronous, one-to-one, and unobtrusive, with ephemeral information. The upshot is that adoption and use of these multiple communication modes by distributed developers would ensure optimum learning about the internal and external environment, and hence optimum information generation and processing. Because information uncertainties are managed through information generation, multiple communication modes should enhance information generation.

Persistence in the face of inter-site information uncertainties, however, induce managers to institute measures that will “enrich” (Daft and Lengel 1986; Lee 1994; Ngwenyama and Lee 1997) information being communicated across
the different sites. Daft and Lengel (1986) argue that information exchanged in technology-mediated communications are generally “poorer” than those exchanged in face-to-face interactions because technology-mediated communications lack cognitive cues such as facial expressions and head movements. However, the absence of cognitive cues as a cause of the poverty of information has been contested by Ngwenyama and Lee (1997) who argue that technology-mediated communications can convey rich information if the interacting parties share a common context. A common context refers to some common phenomenon which serves as a reference for interacting parties by supplementing technology-mediated communication with additional information to enrich the communication. Examples are e-mail and instant messaging archives, document management systems and knowledge repositories. With a shared context, very few words sent through an e-mail communication can even be richer and economical than face-to-face information. For these reasons, managers aim at the conscious development of common or shared contexts that will enhance mutual understanding between distributed developers. Mutual understanding in DSD teams is a learning process supported by psychological contracts between team members (Koh et al. 2004; Lane and Ågerfalk 2007) that should result in shared mental models (Espinosa et al. 2001). The learning process is crucial for the development of shared mental models because team members’ models can remain dissimilar and even lead to a decrease in their interactions (Levesque et al. 2001).

Conflicts

Interpersonal and interunit conflicts within the closed system of an operating core are minimal or reduced by the closure. Conflicts between software developers and between their units are reduced to ensure a high degree of operational certainty and functional interdependencies between all resources. In distributed organizing, however, environmental factors, including different socio-cultural backgrounds and orientations of operating personnel in site B may be introduced into software development operations and engender interpersonal and inter-site conflicts between A and B. For example, different socio-cultural characteristics associated with such locations and with the employees hired may cause problems in information exchange between the sites. Thus, perceptions and motives associated with socio-cultural orientations could induce different interpretations of the same piece of exchanged information by parties in different locations (Sahay et al. 2003). According to Jehn (1995; 1997), two distinct types of conflicts – task-related and relationship – have different bearings on group performance. Task-related conflicts refer to disagreements among group members on the technical aspects of their task. Relationship conflicts refer to interpersonal incompatibilities related more to different perceptions and other socio-cultural characteristics. Task-related conflicts lead potentially to critical evaluation of task-related issues and enhance creativity. Contrarily, relationship conflicts are inversely related to individual and group performance. Drawing on Jehn, in instances where socio-cultural differences between sites A and B lead to task-focused conflicts, developers may turn them into positive performance. But where such differences lead to relationship conflicts, they are likely to persist and undermine problem-solving because they are more difficult to resolve. In this scenario, environmentally-borne conflicts could be additional sources of inter-site conflicts between distributed and interdependent units.

In DSD, the tight coupling between resources and their environment is a pretext for interpersonal and inter-site conflicts to develop. Such conflicts would develop mainly if the socio-cultural backgrounds and orientations of such employees differ from one employee to another and from what the organization is normally used to. These differences may translate into perceptual and motivational differences, and engender interpersonal conflicts (Schmidt and Kochan 1972). To achieve adaptability in the face of such environmentally-borne conflicts, managers ensure continuously that such differences in perceptions translate, at least, into task-focused conflicts and not into relationship conflicts. But knowing that excessive task-focused conflicts lead potentially to relationship conflicts, managers also ensure continuously that such an outcome is prevented. Thus, they promote continuous mutual learning about the socio-cultural norms of the environment from which employees are recruited. Specifically, intercultural learning and orientation sessions which create avenues for informal interactions between distributed developers are held. This is an important strategy for responding to diverse environmentally-borne conflicts.

Persistence through conflict management can be achieved through concentration of environmental sources from which employees are recruited. Relationship conflicts mirror Thompson’s (1960) “latent-role conflicts” which, in his view, should be managed by recruiting employees from a more homogeneous labour force. The heterogeneity or dispersal of labour sources is the basis for excessive differences in socio-cultural characteristics which translate into perceptual differences and latent-role conflicts among employees. Therefore, managers may concentrate such sources to increase socio-cultural homogeneity even among distributed developers. But in the instance of inevitable dispersed employee sources, managers create more homogeneous operating teams to reduce perceptual differences or relationship conflicts among team members.
Technology Representations

Because closed software development operations in collocated organizing are characterized by minimal uncertainties, minimal conflicts and optimal interdependencies, technology representations will normally reflect more of the ideals of technical rationality. Technical rationality is witnessed in a scenario where the specified actions for information technology produce the desired outcomes (Thompson 2003, p.14). However, in distributed organizing, the openness of these relationships to direct environmental influences implies a departure from the ideal of technical rationality towards socio-technical rationality. This is because more non-technical or human resources would be required to complement technology for managing those relationships – “coordination by mutual adjustment” (Thompson 2003).

To understand the coordination functions (representations) of distributed technologies is to understand their roles in enhancing interdependencies, and in reducing conflicts and uncertainties. DSD teams work with technologies such as various information generating systems, information processing systems, and interaction technologies that mediate interactions between the operating core and its environment. Thompson (2003) refers to these as “mediating technologies;” examples are customer relationship management applications, the Internet and electronic data interchange systems. They also work with other technologies that are applied to core operations (“intensive technologies”) as well as “long-linked technologies” that ensure serial interdependencies in the operations (Thompson 2003).

For adaptability through management of technology representations, managers implement easily customizable mediating technologies to deal with diverse and non-standardized environmental stimuli. In the face of diversity of environmental information, mediating technologies must possess capacities for sensing and registering environmental information accurately. These are the foundations for continuous learning about the environment to increase information certainty. Through these tasks, mediating technologies can represent (function as) uncertainty-reducing agents. They can also represent conflict-reducing agents by exploiting environmental information to learn continuously about the socio-cultural norms of the environment and to understand employees’ perceptions and behaviours. Similarly, intensive and long-linked technologies can represent agents that optimize functional interdependencies. This is witnessed in the case where it facilitates exchanges between the distributed sites. Managers aim to achieve this by adopting multiple information technologies that ensure multiple communication modes and offer different yet complementary information and interaction characteristics.

Mediating technologies also aim to implement technologies to represent agents that reduce long-term environmental uncertainties and conflicts, thus ensuring persistence in the environment. This is done by customizing the technologies with mechanisms to process environmental information. In addition, these technologies can represent reducing agents of long-term inter-site uncertainties if they are designed to possess capacities for engendering shared contexts that enrich information.

Figure 2 and Table 1 below summarises the analysis of how the resource relationships perspective enhances analysis of the effects of spatial distribution on DSD coordination.
Figure 2: Conceptual Model of the Resource Relationships Perspective on GSD Coordination
Table 1: Summary of the Management of Resource Relationships

<table>
<thead>
<tr>
<th>Managing Interdependencies</th>
<th>Environmental</th>
<th>Interlocational</th>
<th>Environmental Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptable</td>
<td>Ensure continuous monitoring of environmental dependencies to learn and continuous seeking of alternative dependencies. Ensure continuous tuning and retuning of operations to match dispersal of environmental dependencies</td>
<td>Ensure that timeliness, frequency, contents, volume and variety of information and material exchanges enhance interdependencies.</td>
<td>Environmental dependencies must be dispersed to avoid concentration of dependencies in few sources; OR Contract, coopt or coalesce.</td>
</tr>
<tr>
<td>Persistence</td>
<td>Adaptation, stability and continuity of operations.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Managing Uncertainties</th>
<th>Environmental</th>
<th>Interlocational</th>
<th>Environmental</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td>Ensure continuous learning about the environment.</td>
<td>Adopt multiple communication modes and deploy quality communications technologies. Increase amount of information exchanged per unit of time, number of information representations exchanged, and frequency of exchanges.</td>
<td>Ensure continuous learning about the socio-cultural norms of the environment. Ensure that environmentally-borne conflicts are task-related and not relationship-based.</td>
</tr>
<tr>
<td>Persistence</td>
<td>Adopt mechanisms to generate, filter, categorize and structure environmental information.</td>
<td>Develop common or shared contexts to enrich interlocational communications. Disperse input sources and output destinations</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Managing Conflicts</th>
<th>Environmental</th>
<th>Interlocational</th>
<th>Environmental</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td>Ensure continuous learning about the socio-cultural norms of the environment. Ensure that environmentally-borne conflicts are task-related and not relationship-based.</td>
<td>Increase frequency of information and material exchanges between units.</td>
<td>Source software developers from a homogeneous pool.</td>
</tr>
<tr>
<td>Persistence</td>
<td>Source software developers from a homogeneous pool.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| Managing Technology        | Environmental | Interlocational | Environmental |</p>
<table>
<thead>
<tr>
<th>Representations</th>
<th>Adapt or design and deploy easily customizable (flexible) mediating technologies.</th>
<th>Technologies must be designed, implemented and managed to represent agents for reducing conflicts, uncertainties. Ensure smooth interoperations between long-linked and intensive technologies.</th>
<th>Deploy less flexible mediating technologies. Adopt or design technologies that generate, filter, structure and categorize environmental information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td>Develop less flexible mediating technologies. Adopt or design technologies that generate, filter, structure and categorize environmental information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Persistence</td>
<td>Design or adopt technologies that possess capacities for developing common or shared contexts.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Empirical Example of Managing Resource Relationships: Distributed Development of Emrod

From early March 2006 to late January 2007, Team Emrod, a distributed software development subunit within a multinational information technology organization (represented by a pseudonym, SoftOrg) was upgrading a data mining application (also called Emrod) for remote data collection from its external customers’ servers. This application contributed to the broader application – BigSoft. BigSoft was aimed at supporting SoftOrg’s services to its customers. Several other subunits in SoftOrg (called Release Partners [RPs]) were involved in BigSoft development. SoftOrg hoped to achieve remote connectivity in which automated proactive data mining and diagnosing would occur in external customers’ servers. It also hoped that it would achieve cost reduction by relying on SoftOrg’s expertise around the world. Team Emrod was constituted by twelve developers headed by a project manager (PM): three developers and one architect based in Killarney, Ireland; one support person and one developer based in Watertown, South Dakota, USA; the Technical Lead (TL) and four developers in Bloomington, South Dakota; and one product release manager based in Los Angeles, California, USA. All twelve developers reported to the PM who was also based in Killarney in the same work area with the other four.

Research Design

The study was approached with an interpretive epistemology (Walsham 2006), and a critical realist ontology (Bhaskar 1978; Mingers 2004). An interpretive epistemology was adopted because the software developers, being human, have varying belief systems and interpretations. How they make sense of their social world could not be overlooked. The meanings they would give to their relationships with their task, information, and technology were important sources of understanding. A critical realist ontology, which establishes an independent existing reality, was adopted because spatial distribution of people, task, information, and technology is an objective reality. Moreover, the materiality and functionality of information technology makes it a socially-constructed reality.

Case research strategy was used to explore the relationships between the resources, which relationships were yet unclear. The case strategy does not limit altogether the generality of the proposed perspective because “generality is a property of the necessary relations in real structures” (Tsoukas 1989, p.551) exemplified by technology and spatial distribution in the case. Distance, for example, is an independent causal factor which is not essentially a feature of the empirical domain of the case. Conversely, the relationships between the software development resources were essential features of the real and empirical domains of the case because they are subject to variations. Thus, the case strategy and the critical realist ontology together provide sufficient bases of the transferability of aspects of the proposed perspective.

Multiple data collection methods were combined to produce qualitative evidence, showing the richness of social reality in narratives and notes rather than in numbers. Data was collected through observations (or silent participations) in virtual meetings conducted by the Gamma team, through document and e-mail analyses, through short conversations, through formal interviews, and through one 4-hour face-to-face meeting with the PM. The formal interviews of all the American developers were held at their sites in one week. The rest of the evidence was collected at the Killarney site for the entire application upgrade period (approximately six months). The face-to-face meeting was first, followed by all of document analyses, observations and short conversations concurrently in twenty days out of the six months. These diverse methods were mutually complementary, and were used to ensure the veracity and dependability of the evidence.

Illustration of Interdependencies

The Technical Lead anticipated erratic interdependencies between the remotely-distributed sites. From his experience, he knew that the distance between the sites would negatively affect the interdependent relations between developers in the sites. He feared that there would be high levels of uncertainties between them that could lead to serious conflicts. There were continuous fluctuations in customers’ requirements that presented an unstable task environment to Emrod. Such fluctuations induced further changes in business requirements of RPs. The RPs were operating from sites in India, Brussels, other parts of USA, and Britain. Thus, the team’s work was being affected negatively because these requirements served as inputs for Emrod development. According to the PM, interdependencies
“between Emrod and release partners (RPs) [was] not that good; each partner [had] a different motive; commitment from them [was] not certain; engagement with them [was] continuous but the business requirements [could] be changed by a RP arbitrarily; there [was] competition for shared resources by RPs; interdependencies [were] not smooth at all; business requirements baselines are changing continuously in SoftOrg.”

A more significant fluctuation in Emrod’s task environment was related to the highly critical nature of eleventh-hour changed requirements. In the early days of development, changing requirements were easier to deal with because there were enough time resources at developers’ disposal. However, when the release was approaching, it was more difficult to deal with changing requirements because of the obvious time limitations. The following excerpt, which exemplifies eleventh-hour requirements change, is my transcription of part of a teleconference that was being held three days before the first intermediate release:

The release manager (RM) and the PM become concerned about James, based in India, who has just sent an email to the release manager with a set of new requirements concerning the impending release.

RM: I’m surprised he [James] doesn’t understand what the scope is…. Certainly, we’ve got a lot of issues to be resolved and I don’t know how we’re going to resolve that.
PM: I think we’re good to go.
RM: We need to take them one at a time and get back to them.
PM: It’s the same every time. I’m frustrated. We don’t know upfront what we need to do.
RM: Everything gets up to the last week.
PM: We have the support agreement and so why is all this…?
RM: It’s gonna be an interesting one how we can answer these questions. I think we’re almost guaranteed for rejection. We might get a conditional approval.

Because of the distance between them, the already erratic interdependencies between Team Emrod and the RPs were worsened. For example, their output targets and evaluations were tied to the regulations of those sites, even though they were intermediate products that served as inputs to Emrod development. Team Emrod’s inability to predict the changes in the state of business requirements was a typical instance of direct effects of environment on its inputs.

Managing interdependencies: To achieve adaptation of Emrod development to the spatial distribution of resources, the Technical Lead, who directed team affairs in terms of technical issues, collocated Emrod’s components in the various sites to reduce the interdependencies between them. Thus, he remarked that:

“one of the things I tried to do in terms of task interdependencies as TL is to minimize those interdependencies especially between Killarney and Bloomington and Watertown… I tried to design the tasks so that they are completely independent between the regions. I would not necessarily actually do that if it’s between two engineers on the same site.....”

While this measure reduced uncertainties and conflicts, it led, at the same time, to cross-site information dependencies between the sites. This was because the developers needed to be aware of and relate to the state of each component’s development.

Co-optation was SoftOrg’s default arrangement that was expected to ensure functional interdependencies between Team Emrod and its RPs. However, it was not working as evidenced by the PM’s remarks. Team Emrod neither pursued nor advocated for contracting or coalescing RPs with the aim of reducing the vulnerability of its task to these dependencies. Rather, it opted to remain reactive to them by engaging in highly frequent communications among its members. In order to address the input fluctuations from RPs, the team increased its teleconferences towards release deadlines. These communications exemplify management of interdependencies through continuous monitoring of input sources which is a worthwhile means of adaptability to the environment.

Illustration of Uncertainties

The erratic inter-unit interdependencies constituted uncertainties in the source of inputs for Emrod development because the developers’ coding had to align with RPs’ own to facilitate smooth integration that would make BigSoft a success. Mutual knowledge problems were also typical uncertainties facing Team Emrod. For example, at our first meeting, the PM lamented about “guys making assumptions” in the early days of the project.

Managing uncertainties: Uncertainties engendered by interdependencies required more frequent technology-mediated interactions between Killarney and Bloomington developers. Emrod developers, thus, relied heavily on
technology-mediated communications to achieve mutual awareness of the state of the task at all times. Teleconferencing was predominantly used to reduce uncertainties among the developers because it supported rapid notification of changing requirements, it enhanced mutual awareness of others’ tasks, and it reduced communication redundancy. The frequent teleconferences, conducted in virtual rooms with desktop sharing and instant messaging, did not only ensure continuous adaptability of the teams operations to environmental changes.

This is how a developer explained the team’s response to James’ last-minute requirements:

“In that case, it was mostly emails. Jeff starts an outline. Ok, this is what we need to do, and here’s what everybody’s assigned to do, so go off and do it. And then we would send an update to the whole list; or you just reply-all and say ok I’ve got my part done and here it is. If we had a team meeting scheduled between [the time we learn about the changed requirement and release day], then we would discuss it in our team meeting. We usually didn’t have a scheduled phone conference between the team, just the e-mail – broadcast email [using the team mailing list].”

Team Emrod’s adaptable capacities were complemented by their use of multiple communication modes: e-mailing, telephone calling and instant messaging. These communication modes were applied in various times to match parameters such as the detail of information needed; the reckoned length of the communication; whether the communicator wanted the communication to be obtrusive or unobtrusive; the necessary number of people who needed to get the information being communicated; whether the information needed to be stored or not; and whether the communicated issue required an immediate or delayed response.

The team’s adaptable capacities were found in the high levels of experience and agility exhibited by some of the developers to manage uncertainties. The Bloomington developers had greater experience in remote connectivity applications development in general, and in agile development in particular. Although SoftOrg’s regulations demanded Emrod’s adoption of formal methods which entailed less operational costs, Emrod’s challenges and its capacities for agile development within operational cost limits were crucial for managing uncertainties. The Team Philosophy, for example, read as:

“fast, lightweight, nimble... Do the Right Thing ...at the expense of ‘the process’”

And the Engineering Methodology also read as:

“Our engineering methodology is a combination of a larger, traditional phased approach for use in outward-facing communications, and an internal iterative “agile” methodology for use within the team. The larger methodology is required because we interface with many external organisations that impose this structure upon our team. However, within the team we use an iterative form of the ‘agile’ development methodology.”

Beside adaptability, Team Emrod sought persistence through longevity of developers in the team, and through knowledge repositories: Although the frequently changing requirements close to release time required high agility levels, the challenge also required high degrees of mutual understanding between Emrod’s distributed developers. The developers’ continuous relationship building since the beginning of Emrod development had resulted in high mutual understanding which they exhibited to deal with eleventh-hour changed requirements. For example, when the PM lamented about “guys making assumptions” in the early days of the project, the two Killarney developers added that they had learned continuously about the preferences of Bloomington developers.

This mutual understanding was achieved through continuous relationship building over technology-mediated communications and complemented by few travels. The developers who had met face-to-face through travelling witnessed that those encounters contributed significantly to mutual understanding. This was confirmed by one of the Bloomington-based developers:

“I have [travelled to Killarney before], in fact. I’ve met all of them. It certainly does help to actually know what they look like because now you can put a face to the voice on the phone or a face to the email and say, ‘oh, that’s Gabby’. It makes it a little more personal knowing who’s actually on the other end. Being able to see them adds something. Being able to actually sit across and talk to them whenever possible... it’s good to get together with them so that you can talk about something other than work so that you know that you’re dealing with another human over there. And that usually helps a great deal.”

Nevertheless, only two of the American developers had met the Killarney developers face-to-face. Thus, it can be said that relationship building within technology-mediated communications was the foundation for developing this mutual understanding.
Illustration of Conflicts

There were cultural differences between the Killarney and American developers in terms of manner of conversations, religion, and technology preference. In their technology-mediated interactions, a Killarney developer reported that “we are more social in Killarney,” while the “Americans hone in straight into the task.” He said that the Irish infer a lot from conversations, while the Americans are kind of “black-and-white” in their talking. These differences only resulted in “guys making assumptions” in the early days of Emrod. Beyond those days, their continuous learning had resulted in mutual understanding.

Most of the Americans were Protestants, while most of the Killarney developers were Roman Catholics. However, this religious difference did not present any conflicts in their interactions or work.

Concerning technology culture differences, the Killarney developers preferred and used proprietary technologies for developing their components, while the American’s preferred and used open-source technologies. For example, Killarney developers used Windows Server with two Microsoft® SQL databases, while the American developers used Linux Server with one MySQL® database. In general, the Americans had an open source software mentality, while the Killarney developers had a proprietary software mentality. This cultural difference sometimes caused light conflicts in their work.

Managing conflicts: Thus, the PM was working hard to transfer all their data into an Oracle® database as a means of reducing the technology culture conflicts. The measure that addressed the problem of “guys making assumptions” was his implementation of frequent cross-site interactions between the developers. There was no chance for the PM to manage potential conflicts by recruiting developers from a homogeneous labour pool. The intellectual demands of Emrod development coerced BigSoft to use developers from any of its offices around the world to form the Team Emrod. Thus, the PM had accepted that default, and was depending on the longevity and experience of his developers to manage conflicts. Their longevity built mutual understanding; and their experience helped in dealing with task interdependencies that would have caused conflicts.

Illustration of Technology Representations

If the representation of a problem is understood as a misrepresentation, then the differences in technology preferences can also be understood as instances of technology misrepresentations. This is because the open-source and proprietary technologies preferred by developers were representing conflict-enhancing agents instead of the reverse which Team Enrod desired. There were therefore some misrepresentations of the team’s intensive technologies.

The reality of “guys making assumptions” in their technology-mediated interactions indicates how ‘poor’ the technology media they used were. This description of technology reflects the ideals of technical rationality. The technology-mediated interactions were without cognitive cues such as facial expressions and head movements leading to many assumptions that caused mutual knowledge problems. The team’s long-linked technologies therefore exhibited some representations of agents that enhanced uncertainties.

Managing technology representations: The PM’s ambition to transfer the team’s data from the diverse databases into an Oracle® database was aimed at managing the intensive technologies to represent conflict-reducing agents. At least, it would remove the conflicts that were engendered by technology culture differences. The aim was long-term and reflects an orientation towards persistence in the face of cross-site cultural differences.

The PM addressed the mutual knowledge by implementing different kinds of information and communication technologies to give the developers sufficient interaction options. Thus, there was variety in long-linked technologies that was requisite to address the differences in the developers’ communication preferences. This requisite variety, coupled with the frequent interactions and longevity of developers, reflected the ideals of socio-technical reality. This reflects a move to ensure the adaptability of the team’s operations to fluctuations in preferences for interaction media. The information exchanged by them was enriched significantly because they were achieving high levels of mutual understanding. By the time of my empirical research, the team had used some of these technologies to store some of their interactions. This gave it a huge repository of knowledge that served as a common context to enrich information and enhance mutual understanding. Thus, the long-linked technologies were being managed to represent agents that were reducing uncertainties. Furthermore, the management reflects a long-term aim for persistence of the team’s operations against cross-site uncertainties.
To avoid the problem where the team’s technologies would represent agents that undermined environmental interdependencies the PM deployed automated remote connectivity systems that reported on the circumstances of already installed technologies. These systems are examples of mediating technologies. But they were not isolated. They were combined with a well-organised human-based support system and the team’s frequent interactions to constitute a functional socio-technical system. Thus, both sets of customer queries and changing requirements could be managed with the help of the team’s mediating technologies. These technologies and the accompanying measures were typical examples of how to manage technologies to represent agents that address erratic interdependencies. They also reflect measures that ensured adaptability of the team’s operations to fluctuating environmental interdependencies.

These Emrod examples show that relationships between software development resources, which require management to ensure sound coordination, are significantly affected by spatial distribution. The effects are diverse, and indicate that there may be more of them to be identified in other DSD contexts. The examples also show the management interventions implemented by the PM to ensure adaptability and persistence in the face of loose coupling between the resources. It is clear from the examples that all the resource relationships – interdependencies, conflicts, uncertainties, and technology representations – together serve as a creditable perspective for analysing the spatial effects on coordinating DSD.

**Discussion and Implications of the Resource Relationships Perspective for the Analysis of Coordination in DSD**

This perspective complements existing ones, yet it is a distinctive contribution to research in IS development because of three strategic benefits it brings. First, it leads to more systemic and unified analyses of DSD coordination because it induces the analyst to account for all the resources, and to study the dynamic interrelations between them. This understanding is an important precursor to conceptualizing how software development operations can adapt and persist in the face of greater exposure to the environment. Existing perspectives do not provide systemic explanations because they focus researchers’ attention either on the resources or on the analysis of individual resource relations. For example, Herbsleb and colleagues (2001; 2000) and Herbsleb and Mockus (2003) approach coordination of DSD focus on how distance affects speed and delay in communications.

Second, it instigates greater in-depth analysis of DSD coordination because resource relationships are the very basic expressions in software development operations. Moreover, they are not merely presented in this paper as isolated expressions but as interrelated ones. This means that the adoption of this perspective leads to a meta-relationships – relationships of relationships – analysis. The greater depth of analysis afforded by this perspective will help analysts to obtain greater accuracy in their statements of the resource relationships and how they are affected by spatial distribution. For example, as different distances between distributed resources may engender different meta-relationships, this perspective can effectively configure these different relationships more accurately. Thus, it is the most effective means of accounting for the experience of DSD coordination.

Third, it directs researchers’ attention to ongoing processes of software development where relationships are enacted and adjusted. It therefore leads to a more phenomenological analysis of DSD coordination. It represents a significant departure from perspectives that direct attention to the individual objectified resources and to how spatial distribution enhances or debilitates their application in software development operations. However, this perspective focuses on understanding the domain of phenomenological relationships that spring from interactions between spatial distribution, information, people, technology, and the software development task. Therefore, appreciating these resource relationships in DSD must be the analyst’s central focus as he or she considers the following implications.

Interdependence, perceived as a relationship, is a continuous variable. Therefore analysis of degrees and variations in degrees of interdependencies over the course of development is important. In analysis, all the interrogative pronouns – what (management actions), how (methods), where (locations of actions), when (times of actions), and under what circumstance (contexts) – may be applied to explain how interdependencies are managed, and how they are shaped by other relationships.

The consequences of conflicts in DSD are as important as the antecedents because there is the likelihood of circular causation between them. Antecedents and consequences may reinforce existing conflicts, transmute them, or generate entirely new ones if management is not directed at both of them. Some people’s motives may be directly
linked with the socio-cultural norms of the particular locations in which they work, or where they have been nurtured. For example, Walsham’s (2001) analysis of cultural-based structural contradictions and conflicts between globally-distributed software teams attests to this reality. Conflict may also be linked with one party’s perception of an opportunity for interference by another party. Perceived opportunity for interference is a potential occurrence in scenarios where the two parties share a common resource.

The analyst must also explore how uncertainties are managed to ensure certainty of the state, cause or effect of a phenomenon (Milliken 1987). Uncertainty is an entity’s inability to predict information about a phenomenon accurately. Closely related to uncertainty are equivocality, which refers, to the ambiguity of some information and its sources as perceived by an entity (Daft and Lengel 1986; Milliken 1987 p.136). The interesting feature of the relationship between uncertainty and equivocality is that they relate in cyclical causation. Information generation as a measure for managing uncertainties may lead to new equivocalities, which management through information processing may cause entropy and new uncertainties, which may require further information generation, and so on. Therefore, it is important for the analyst to consider the management of this cyclical causation in relation to distance, socio-culture and technology.

It is important to understand technology and their representations as tools and signs that, according to Vygotsky (1978), are mutually interrelated and separate at the same time:

“[The] most essential difference between sign and tool, and the basic real divergence of the two lines, is the different ways that they orient human behaviour. The tool’s function is to serve as the conductor of human influence on the object of activity; it is externally oriented; it must lead to changes in objects. … The sign, on the other hand, changes nothing in the object of a psychological operation. It is a means of internal activity aimed at mastering oneself; the sign is internally oriented.” (p.55) (italics in the original).

Thus, on the one hand, the tool representation must be understood in terms of how it facilitates people’s efforts in a software development activity (external orientation of technology representations). On the other, the sign representation must be understood in terms of how it shapes people’s psychological attributes such as attitudes, feelings, perceptions, motives and frames of reference. These attributes are important for understanding the antecedents of conflict, the potential roles of equivocal and uncertain information, and the management of technology representations.

**Concluding Remarks**

This paper proposes the resource relationships perspective on analysing the spatial effects on DSD coordination. It does this by showing how the resource relationships constitute a more systemic, in-depth, and phenomenological perspective. The effects of spatial distribution on the resources have been explained to show why they lead to loose coupling in-between them, and to tight coupling between them and the environment. The organizational responses to loosely coupled systems – persistence and adaptability – are used to conceptualize the resource relationships perspective. The conceptualization has been illustrated with an empirical example of software development distributed between sites in USA and Republic of Ireland. Through this conceptualization, the paper explains why coordination of DSD projects cannot be perceived only in terms of any one of the resource relationships. It draws attention to their inseparability and calls for analysts to pay attention to how one affects the other in their empirical studies. Therefore, it has underscored the resource relationships perspective as a complement to the resource dependency and transactions cost perspectives on DSD coordination that are currently in use.

**Acknowledgements**

This research was conducted in the Social, Organizational and Cultural aspects of Global Software Development (socGSD) project at the University of Limerick, Republic of Ireland. socGSD was one of the LERO (the Irish Software Engineering Research Institute) cluster projects funded under PI grant 03/IN3/1408C by Science Foundation of Ireland (SFI). I am grateful my socGSD colleagues, Gabriela Avram, Daniel Sullivan, Michael Hales, Anne Sheehan, Anders Sigfridsson, and our leader, Liam Bannon, for their help in developing the seminal ideas of this paper during the 2 years I spent with them. I am also grateful to the informants in the studied organization who allowed access to them throughout the study period.

[Thirty First International Conference on Information Systems, St. Louis 2010](#)
References


