ABSTRACT

In this paper, we conceptualize the relationships between software developers’ socio-cultural differences, global distribution, intercultural learning and interpretation of information in the organization of GSD. Our discussions are centred on the idea that the combination of global distribution and different socio-culturally-based perceptions will lead to interlocutors’ heterogeneous interpretation information. Such interpretation contributes largely to several organizing problems that have been reported in the literature on distributed work – e.g. conflicts, uncertainties, misattributions, ethnocentrism and mistrust. We propose intercultural learning as a strategic response for dealing with different perceptions and hence for moderating the heterogeneity of information interpretation. On the methods to attain intercultural learning, we propose the creation of avenues for informal interactions, the increment of the frequency of interactions, and the creation of common contexts for enriching information exchanged in remote interactions.

Keywords: Global software development, intercultural learning, socio-cultural, perception, interpretation, information.
INTRODUCTION

The ‘global’ of global software development (GSD) suggests the geographical or remote distribution of development efforts across, at least two, countries around the world. Carmel (1999) talked about the deployment of best expertise, development and time-to-market costs reduction, global presence, and proximity to customers as some of the key motives that drive software companies to distribute development across borders. However, emerging research results from field studies on GSD indicate that in spite of the achievement of some of these motives, they usually come with socio-culturally-borne problems which sometimes overwhelm the cost savings (see, for example, Cramton 2001, 2002, Grinter, Herbsleb, & Perry 1999, Herbsleb & Grinter 1999, Hinds & Bailey 2003, Krishna, Sahay, & Walsham 2004, Sahay, Nicholson, & Krishna 2003). Existing literature on GSD points out that socio-cultural differences between developers working in different global locations are, arguably, the greatest problems confronting this new mode of software development. Thus, the combination of socio-cultural differences and global distribution constitutes a significant challenge for organizing GSD. One manifestation of this challenge occurs in remote interactions, where the information exchanged is interpreted differently as a result of different socio-cultural backgrounds and predispositions of the interlocutors.

Since software development aims at the developing workable code – intangible products that contrast with tangible products of other types of work – the development process is dominated by information. Such pieces of information manifest in representations such as text, voice and images; and their generation, processing, storage and transmission constitute the majority of developers’ actions. Although it is well known that code development is done mostly in isolation because of the amount of concentration needed by the developer, it is also well known that
developers work in teams, signifying a high magnitude of task interdependencies between them. Thus, it is also necessary for them to interact to exchange information to accomplish intermediate and ultimate outcomes. It is in these exchanges that heterogeneous interpretations of information are potential; and we premise this paper on the idea that heterogeneous interpretation of information will manifest in interactions between developers who are characterized by different socio-cultural backgrounds and predispositions. Several organizing problems seem to result from heterogeneous interpretation of information in GSD, and, therefore, propositions of strategic responses to address this socio-cultural challenge would also help in moderating the heterogeneity of information interpretation and in resolving the corollary organizing problems.

In this paper, we propose intercultural learning as a strategic response to the socio-cultural challenge which is pervasive in GSD. We define “intercultural learning” as interlocutors’ continuous mutual awareness of and adaptation to each other’s socio-cultural backgrounds and predispositions. That intercultural learning helps in managing the adverse effects of socio-cultural differences is not new because intercultural communications and adaptation are commonplace themes in the Communications and International Relations literatures (see, for example, Beraldi 2006, Clyne 1994, Kim 2001). The distinctiveness of our conceptualization, therefore, lies, first, in our explanation of how intercultural learning will help in moderating heterogeneous interpretation of information within the context of GSD; and second, in our proposition of methods of intercultural learning within the context. That is to say, we back our propositions with discussions of how learning can help in the moderation of heterogeneous information interpretation, and hence in the enhancement remote interactions. We propose three methods: creation of avenues for informal interactions; increment of the frequency of
interactions (formal and informal); and creation of common contexts for enriching information exchanged in technology-mediated interactions.

In our discussion, we establish relationships between socio-culturally-based perceptions, global distribution, intercultural learning and developers’ interpretation of information in GSD (see Figure 1).

We will then discuss how interactions, learning, adaptation and awareness can shape and reshape perceptions and attitudes, leading to more efficient organization of GSD. We argue, eventually, that for optimal organizing of GSD, intercultural learning as a strategic response, which ensures that globally-distributed software developers’ interactions, awareness, and adaptation together orient their perceptions and attitudes positively, is a fundamental imperative for optimal organization of GSD.

We believe our discussion is important and timely on the grounds that socio-cultural problems are being reported continuously as the key bedevilling challenge of GSD by both researchers and practitioners (see, for example, Krishna et al. 2004, Sahay et al. 2003, Walsham 2001). Interpersonal and task-related conflicts as well as information and task uncertainties in software development activities are some of the main negative consequences of socio-cultural
differences between distributed developers. Although these problems are well acknowledged by most GSD researchers, very little attention has been devoted to analyzing the mutual relationship between socio-culture and distributed developers’ continuous intercultural learning, awareness and adaptation; and to how the proper management of this mutual relationship can lead, for example, to better coordination, collaboration and knowledge sharing in GSD. Through our analysis, therefore, we aspire to bring greater understanding to these relationships and instigate further research in the areas we will touch on, and on other challenges that relate to socio-cultural differences in GSD.

GLOBAL DISTRIBUTION AND SOCIO-CULTURAL DIFFERENCES

In view of the fact that “distance matters” (Olson & Olson 2000), the global distribution of software development will be further encumbered with organizing the concerted efforts of developers who are separated by various types of distance. For instance, in her study of “knowing in practice” in distributed software development, Orlikowski (2002) discerned seven types of boundaries (distance) – temporal, geographical, political, technical, cultural, historical and social – that shaped and challenged distributed work. In this paper, our discussion will implicitly involve all these types of distance; however, we will place more emphasis on the cultural and spatial types of distance because we seek to illuminate the socio-cultural differences inherent in spatially-distributed software development.

A person’s socio-culture may be a corollary of his or her familial upbringing, professional training, environmental influence, religion, tribe, or any combination of these factors. The corollaries are mental characteristics such as perception, motive and interest; and they manifest in critical behavioural characteristics such as attitude, mode and style with regard to actions
Our propositions are premised on the idea that an individual’s perception and attitude will shape his or her interpretation of information – a determining variable that conditions the outcome of interactions, and hence, the organization of software development across borders. In other words, the socio-cultural challenge of GSD organization stems from the idea that perceptions and attitudes are not static characteristics. The dynamics of an individual’s perception and attitude suggest that they will be shaped and reshaped continuously over the course of a developer’s engagement with software development work; and that they will further shape and reshape developers’ interpretation of information, and, ultimately, their willingness to share information and knowledge (Leidner 2001). We posit that this continuous shaping and reshaping will occur as developers in different locations interact with, learn about, become aware of, and adapt to each other’s perception and attitude.

The lifeblood of interactions is information, and the way it can be interpreted differently by different developers according to their socio-cultural orientations illuminates how problematic the interactions between such developers can be for organizing distributed development. We ground our discussions on our submission that the interpretation of any piece of information exchanged in interaction is always a factor of both socio-culture and spatial distance. In the following sections, we examine the consequences of our submission in detail by making propositions on the specific relationships between these factors and on strategies for tackling the socio-cultural challenges.

**Interpretation of Information**

Research publications on culture and organizational behavior usually espouse the individual-collective dichotomy as most influential in human behaviour (Gudykunst et al. 1996, Gudykunst 1997, Leont'ev 1978).
& Nishida 1986, Hofstede 1997, Singelis & Brown 1995). In this body of research, there is the notion that a mutual relationship exists between the individual and his or her culture; and thus an individual’s behaviour is an outcome of how much he or she is influenced by culture (see, for example, Singelis et al. 1995). The communication behavioural patterns of software developers who are globally distributed, we would surmise, are factors of the cultures that dominate their perceptions and attitudes. In terms of distributed software development, this suggests that the behavioural outcomes of socio-culture will manifest in the remote interactions between any two developers. And therefore, socio-cultural differences between two interacting parties will manifest in differences in each party’s interpretation of the information exchanged between them.

We have already alluded to the idea that factors like familial upbringing, religion and tribe are the determinants of a person’s perception and associated attitude. In this discussion, we conceptualize information interpretation to lie on a continuum between professional and non-professional cultures. Professional culture signifies software developers’ perceptions and attitudes that reflect the software development profession and the near-universal professional training that is associated with it; while non-professional culture reflects their perceptions and attitudes that derive from familial, tribal, religious orientations, and/or any other attribute that is not professionally-related.

The typical scenario in GSD is that developers who, in most instances, have not met each other face-to-face before, must interact remotely to collaborate in software development. The challenges posed by socio-culture within these interactions can be conceptualized, based on the assumption of an interaction between two developers, in terms of the extent to which each one’s perception is more (or less) induced by professional or non-professional culture. Based on this
conceptualization (in Figure 2), the only interaction scenario in which homogeneous interpretation of information is most likely is when distributed developers are both more influenced by their professional cultures. This is the ideal interaction scenario. Any other interaction scenario is likely to result in differences in information interpretation; and the scenario in which the worst form of heterogeneous information interpretation is most probable is in the interaction between developers who are both more influenced by their non-professional cultures.

![Figure 2: Consequences of socio-cultural differences](image)

In view of the fact that our discussions in this paper are centred on the implications of and strategic responses to heterogeneous interpretations of information, we shall assume the worst case scenario in which both interacting developers are more influenced by their non-professional cultures. Note that we do not imply equality in socio-cultural backgrounds and orientations when we say that both are more influenced by their non-professional cultures. On the contrary, we presume each party’s perception to be oriented by an entirely different form of non-professional culture.
Organizing Problems

When socio-cultural difference couples with distance (global distribution) between the interacting parties, necessitating technology-mediated interactions, we would expect the heterogeneity in interpretation of information to be magnified or more difficult to manage. Research on distributed organizing and communications explicate the consequences of heterogeneous interpretation of information in terms various organizational problems (see Figure 3). For example, Krauss and Fussell (1990), in their study of communication effectiveness, and Cramton (2001), in her study of dispersed collaboration, both suggested that lack of mutual knowledge as an organizational consequence. Hinds and Bailey (2003), in their study of communication between distributed teams, reported that being “out of sight” and “out of sync” with others combines with differences in information interpretation to result in conflicts. Cramton (2002) argued that attribution errors (misattributions) among dispersed collaborators would be considerably greater than among collocated collaborators due to interpretation differences. Jarvenpaa and Leidner (1999), and Jarvenpaa and associates (2004), in their study of communications in global virtual teams, underlined trust (or mistrust) as crucial, and suggested that mistrust is a central consequence of interpretation differences in virtual teamwork. And Cramton and Hinds (2005) conceived ethnocentricism to result potentially from remote interactions dominated by cultural differences between distributed teams.

Figure 3: Organizing consequences of heterogeneous interpretation of information
Looking at these organizing problems, it is conceivable that unwillingness to share information and knowledge on the part of distributed workers will be a further consequence. The main challenge facing this paper, therefore, can be framed as: how can the organizing problems resulting from socio-cultural differences between globally-distributed software developers be managed? We suggest that intercultural learning is a worthy strategic response to this question.

INTERCULTURAL LEARNING

Since people’s cultural backgrounds and orientations cannot be changed in the short term, as in the period of a specific software development project, project managers must direct their efforts towards facilitating intercultural learning. Thus, while individual cultures may remain different, intercultural learning can reduce the heterogeneity of information interpretation in remote interactions between developers. Lessons from intercultural learning in collocated yet culturally diverse work groups give credence to the idea that learning and adaptation can be achieved in the face of cultural diversity (see, for example, Cramton et al. 2005, Ely & Thomas 2001). We propose that intercultural learning between distributed developers increases each developer’s awareness of and adaptation to the other’s perception and attitude. Thus, through the same interactions which problematic nature results in organizing problems, learning, adaptation and awareness can be achieved.

For instance, in his study of experiences of a distributed organization, Abel (1990) observed that learning about work colleagues in other locations was essential for enhancing and fostering more understandable interactions between such parties subsequently. Learning signifies a continuous and conscious process of knowing that and knowing how (see, for example, Cook & Brown 1999, Garud 1997, Orlikowski 2002). ‘Knowing that’ is a way or achieving awareness
while ‘knowing how’ is essential for adaptation – and the two are mutually reinforcing. Knowing
the perception and attitude of a developer in another location is a way of knowing how to interact
with the person; and knowing how to interact with that developer is a sound basis for increasing
awareness of his or her perception and attitude. Awareness fosters adjustment and hence mutual
adjustment which, according to Thompson (2003), is a valuable coordination process for dealing
with unpredictable situations as in the outcomes of the clash of socio-cultural idiosyncrasies. On
the other hand, adaptation fosters tolerance which is a necessity for the achievement of mutual
adjustment.

Let us assume that mutual adjustment is tantamount to modified or nullified perceptions and
attitudes; and that such modification or nullification has moderated successfully the
heterogeneity of information interpretation. Since human beings are largely unpredictable, and
since adaptation and awareness are continuous processes, we should expect continuous
interpretation challenges to confront remote interactions. Thus, we would expect awareness and
adaptation on the one hand and the modification or nullification of perceptions and attitudes on
the other to relate in cyclical causation (see Figure 1). In other words, the moderation of
heterogeneous interpretation of information will be provisional and will always be subject to
what extent interacting parties have adjusted to and tolerate each other’s perception and attitude.
Thus, subjection is key because of other factors that can undermine mutual adjustment and
necessitate its renewal.

This brings into question what managers must do to remote interactions to facilitate
intercultural learning as a way of moderating the heterogeneity of information interpretation and
of resolving the organizing problems. We suggest that creation of avenues for informal
interactions, an increase in the frequency of interactions (formal and informal), and the creation
of common contexts for enriching information exchanged in technology-mediated interactions are worthwhile strategies for enhancing intercultural learning.

**Informal Interactions**

One essential ingredient that fosters learning and adaptation among culturally-diverse individuals is informal interactions. The typical scenario in collocated software development projects depicts both formal and informal interactions from the beginning of a project. Thus associated factors such as proximity, and sporadic and frequent interactions (both formal and informal) enhance intercultural learning significantly. In contrast, remote interactions in GSD projects would be typically formal at the outset. And when it happens that distributed developers have socio-cultural differences that result in heterogeneous interpretations of information, a strain is put on collaboration and coordination. It is important, therefore, that arrangements are made to facilitate, as much as possible, informal interactions between distributed developers in the organization of distributed software development.

We propose, based on social psychological accounts of information richness in human interactions (Daft & Lengel 1986, Short, Williams, & Christie 1976), that arranging early face-to-face encounters between distributed developers is key to the achievement of mutual understanding between such developers. In these arrangements, even if meeting agendas are project-related and hence formal, the encounter will automatically engender avenues for socialization and informal interactions. And through both formal and informal face-to-face interactions, developers are likely to learn few, yet critical, aspects of their colleagues’ perceptions and attitudes. The promise of face-to-face encounters lies in the foundation it lays for smoother and better learning experience during subsequent remote interactions. It promises
virtues such as greater mutual tolerance and patience in subsequent remote interactions; and these are necessary for intercultural learning. This suggests that optimum moderation of heterogeneous interpretation of information can be attained when face-to-face encounters are arranged at the beginning of distributed software projects.

**Frequency of Interactions**

An increase (rather than a decrease) in the frequency of interactions is another method for intercultural learning. We argue that the height of frequency of information exchanges between distributed developers would have bearings on each one’s learning and awareness of the other’s perception and attitude. If the frequency is high, then the frequency of opportunities for learning and awareness would be high, and vice versa, because information exchange can be exploited by interacting individuals to increase knowledge and understanding of each other. This means that highly frequent interactions (both formal and informal) can lead to continuous and increased awareness of the perceptions of remote developers on the one hand, and to appreciation of other cultures on the other. The underlying assumption is that low frequency of interactions can lead to the reification, entrenchment or normalization of perceptions (Wiredu 2006). In other words, if interactions frequency is low, then perceptual differences can last longer and be reified, entrenched or normalized in a developer’s mental frame and operations before the other can be aware of them for resolution. The normalization of different perceptions over a long period can lead to irresolvable or difficult-to-resolve conflicts when the time comes for such developers to harmonize their separate operations through interactions. Since such perceptual differences are major potential causes of interlocational conflicts, and since their reification, entrenchment or normalization requires relatively more efforts and resources to resolve, we propose that it is
better to have a high frequency of information exchange for conflict resolution. In short, conflicts between distributed developers can be decreased by increasing the frequency of interactions.

Common Contexts

The third intercultural learning method is the development of common contexts out of interactions. A common context refers to some common phenomenon which serves as a reference for interlocutors by supplementing technology-mediated communication with additional information to enrich the communication. An e-mail archive is an example of a common context. The underlying assumption is that, all things being equal, information exchanged in technology-mediated interactions are “poorer” than, say, face-to-face interactions – the latter is presumed to offer more complete (or “richer”) information to interacting parties than the former (see, for example, Daft et al. 1986, Short et al. 1976, Whittaker 2003). And drawing upon the arguments of Lee (1994) and Ngwenyama and Lee (1997), the only way information exchanged in any technology-mediated interaction – e.g. e-mail – can be enriched is through the existence (or creation) of a common “context of meaning” that will fill the information gaps resulting from absence of the face-to-face facet of interaction. Common contexts serve as references which can be drawn upon to understand another person’s communication better. They can be either external (physical) or internal (mental); but their essence lies in the idea that they lend themselves to be drawn upon by an interlocutor for understanding another’s perception and attitude.
CONCLUSION

We are aware that the practical challenge in software development does not make learning through interactions a straightforward issue; and that our model appears simplistic and general. In software development, the interaction demands vary from stage to stage in the development process; and, certainly, there are some stages (such as coding) in which interactions between developers would be lower than others (such as testing). To relate our construct to these nuances of GSD is to be accurate with specific details in the analysis; however, little accuracy can be achieved without an empirical component. In this regard, our propositions are, undoubtedly, speculative as of now; and that is the major limitation of our model. This weakness is reflected in the postulate that a model of human behaviour cannot be general, simple and accurate at the same time (Thorngate 1976, Weick 1979: 35-39). It is very clear that the accuracy of our propositions will require detailed empirical investigation and validation (and even that could potentially offset generality or simplicity). Thus, while we accept the limitation, we also acknowledge the inevitable trade-off that comes with any conceptualization of human behaviour; and leave our model to further empirical validation and scrutiny.

REFERENCES


