Shared Understanding and the Effects of Culture in the Global Software Development Team – A Case Study

by

Yvonne Ying-Fan Hsieh

B.A.Sc., University of British Columbia, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate Studies

(Electrical and Computer Engineering)

The University of British Columbia

April 2007

ABSTRACT

This thesis describes a qualitative case study whose goal is to characterize certain aspects of shared understanding among members of a globally distributed software development team. The research questions upon which the study is based investigate the effects of intercultural differences on the development of shared understanding in the team. Specifically, the study examines the developers' shared understanding with respect to the development processes and practices, system requirements, technical details, project scheduling, resource management, and a number of other task- and team-related issues. With regard to intercultural differences, the study focuses on the concepts of risks, hierarchy, time, and teamwork, expression of emotion, and communication patterns. The study data is collected through semi-structured interviews and analyzed using a constant-comparison approach. The findings, summarized as a set of propositions, show that intercultural differences, along with other contextual factors (i.e., communication mechanisms and project arrangement) do affect shared understanding in the distributed team. The work described in the thesis helps to provide insight into the coordination and management of distributed software projects, as well as to further an area of research where little previous work has been done.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank several people for providing me with guidance and support, without which I might not have been able to complete this thesis.

First to my advisor, Philippe, I truly express my heartfelt gratitude for all your wisdom, patience, and encouragement in these past few years. You have continuously inspired me by offering sharp insights and by showing so much enthusiasm about research and software development. At many moments of doubt and frustration, you have always been my side to encourage me to keep moving forward, and I thank you so much for that.

To Eve, it was great to have been able to share this journey with you. You have been an incredible friend and mentor. I would not have been able to finish this thesis without you.

To all my fellow SEALs, David, Larix, Mandana, Steve, Jaana, Davide, and Agung, thank you for all the support, encouragement, and resources you have provided me with in these past years. You guys have been great!

Finally and most importantly, to Mom and Dad, thank you for supporting me emotionally and financially through these long years of education.

CO-AUTHORSHIP STATEMENT

The study described in this thesis is conducted in collaboration with Eve MacGregor of the UBC Software Engineering and Architecture Laboratory during the period 2004-2007. The collaboration is mostly during the data collection phase of this thesis study (as described in Chapter 3). The author of this thesis is responsible for all the data analysis presented herein.

CHAPTER 1
INTRODUCTION

Coordination has long been identified as a major challenge in software development. The challenge is further exacerbated when the development project is distributed across geographical boundaries. Software researchers and professionals have strived to study and control all the factors that might affect coordination in the distributed project. While much of the effort has been directed to study such factors as physical distance and time differences, with the goal of developing new collaborative technologies and development or management techniques, the study described in this thesis explores the more social and transient aspect of coordination — the concept of *shared understanding* in the distributed development team. Research shows that shared understanding among team members is beneficial to team performance, especially under conditions where non-routine tasks are involved and communication is limited (as is the case with the distributed software development team). Shared understanding helps developers make accurate explanations and predictions of tasks and actions, which in turn facilitates coordination. The contribution of this thesis is a set of propositions characterizing shared understanding in the distributed development team. This is therefore an exploratory, rather than confirmatory, empirical study.

Another motivation for this study is the issue of *cultural diversity* and how it can affect the performance of the distributed development team. As the team is dispersed across geographical locations, extensive (and often virtual) collaboration is required of developers of diverse cultural backgrounds. These developers often differ drastically in their attitudes and approaches towards

software development. Such differences, if left unmanaged, might create misunderstandings, conflicts, or underutilization of talents, ultimately diminishing team performance.

Furthermore, this study recognizes that culture is *dynamic* rather than static. In other words, intercultural differences may be bridged and managed over time, as developers gain a deeper appreciation for others' values and approaches. Following this premise, this study also investigates strategies that software professionals may use to establish shared understanding in their distributed, intercultural development team.

The work described in this thesis examines the effects of cultural diversity on shared understanding in the distributed software development team. The constructs investigated in this study are derived from literature on shared cognition, cultural research, and empirical software engineering. These constructs cover dimensions of culture (e.g., hierarchy, sense of time, communication patterns) and aspects of shared cognition (e.g., task-specific knowledge, task-related knowledge, values and attitudes). Moreover, the study takes a "grounded" stance in the sense that several emerging, intervening constructs are considered. These constructs are not the primary concern of the study but emerge from the data as bearing significance to the research questions.

I use qualitative research methods in the study. Specifically, I conduct a case study on a large-scale global software development project. Qualitative methods are chosen because both shared understanding and cultural diversity in global software development are relatively unexplored research areas, with no well-grounded theories or hypotheses. Furthermore, qualitative research

methods allow me to flexibly explore a topic that involves a multitude of contextual factors, as well as to provide detailed insights that may support project personnel in the planning and risk management of distributed projects.

In the following sections, a description of the specific problem addressed by this study is presented as well as the research questions and the organization of this thesis.

1.1 Problem Statement

Coordination is a major challenge in the globally distributed software development project. There is often a lack of visibility into the activities, processes, status, resources, and needs at the remote development sites. Current software development and management methods are less than effective for managing the distributed project. These methods focus on the identification and completion of individual artifacts but overlook the importance of promoting common, transient knowledge and understanding in the distributed team.

Cultural diversity has been identified as an important factor in global software development. To-date, however, there are few studies that systematically and empirically analyze the effects of culture on distributed software development.

We do not currently know how shared understanding is developed and maintained in the distributed development team. We do not know whether and how cultural diversity affects the development of shared understanding. Lacking knowledge of these issues, the developers may attribute inappropriate causality for the challenges and problems in the project. Additionally, we can not establish effective development and management practices for the distributed software

development project until we have a better understanding of the important phenomena and factors present in the distributed environment.

1.2 Research Questions

Current study of coordination, shared understanding, and effects of cultural diversity in software development is not advanced to the point where it is possible or meaningful to test specific, well-founded theories or hypotheses. As such, the work described in this thesis is based on the following exploratory questions:

1. What is the level of shared understanding in the globally distributed software development team, both as perceived by the developers and as evidenced by the team performance?

2. How is shared understanding developed in the globally distributed software development team?

3. How does cultural diversity affect (positively or negatively) the development of shared understanding in the globally distributed software development team?

Shared understanding, here, describes the extent to which the distributed developers are aligned in their knowledge, attitudes, approaches, and goals regarding the various activities and issues involved in the development project. Specifically, this study investigates the team's shared understanding with regard to the following issues: development processes and practices, system requirements, technical details, scheduling, resource management, and other task-/team-related issues. To investigate the effects of intercultural differences on shared understanding, the study

draws from literature on cultural research and adopts a set of cultural dimensions, including hierarchy, sense of time, communication context, collectivism/individualism, uncertainty avoidance, and expression of emotions. Moreover, as this is an exploratory study, it remains open to the possibility of emerging constructs. The emerging constructs may augment the predefined constructs listed above, or they may be intervening factors that affect shared understanding in the distributed development team, but they are not the primary concern of this study.

1.3 Organization of this Thesis

The rest of this thesis is organized as follows: A discussion of current literature on empirical software engineering, coordination, shared cognition, and cultural research is first presented in Chapter 2. Chapter 3 discusses the research methods used in the study. Findings and resulting implications are discussed in Chapter 4 and Chapter 5, respectively, followed by the conclusion and recommendations for future research.

CHAPTER 2

REVIEW OF THE LITERATURE

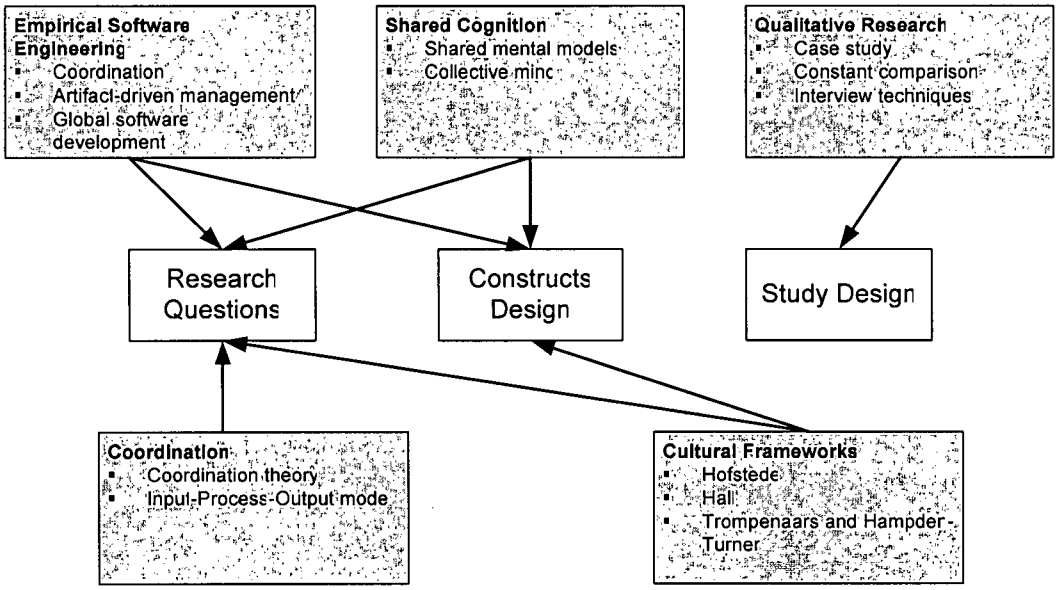


Figure 2-1 Summary of Supporting Literature

The literature reviewed in the following sections contributes to this study by informing the research questions, the design of the constructs, and the study design (as outlined in Figure 2-1). The research questions that this study addresses are informed by themes present in studies of coordination and are further shaped by literature on cultural research, shared cognition, and empirical software engineering. The choice and design of the constructs investigated in the study are also influenced by research on culture, shared cognition, and empirical studies of software development. The specific study design is guided by qualitative empirical research methods.

These five areas of literature are discussed in the sections below in terms of the themes relevant to this thesis and their specific contributions to the different parts of the study.

2.1 Coordination

Coordination is a problem inherent in the work of any large-scale organization. The topic has been studied by many researchers, in different disciplines and from different perspectives. While many view coordination as an issue of managing dependencies [1-3], others discuss it in terms of information uncertainty [4] and interpersonal and inter-unit conflicts [5, 6]. Researchers also differ in their analysis of the type of system in which coordination takes place: some focus on computer systems; others consider human or social systems; and yet others focus their discussion around more complex systems involving both computers and humans [1]. Literature on coordination informs this study's research questions. By highlighting the limits of coordination processes commonly used in organizations including the software development team, this volume of research motivates me to explore the other more social, cultural aspects of coordination.

2.1.1 Existing Theories on Coordination

The most quoted theoretical framework on coordination is perhaps Malone and Crowston's coordination theory [1]. The theory, drawing from a variety of disciplines including computer science, organization theory, economics, and psychology, analyzes coordination in terms of *task interdependencies*. Malone and Crowston define coordination as "managing dependencies." They view the problem of coordination as arising from "actors performing interdependent tasks" that "require or create resources" in the process [7]. Following this premise, Malone and Crowston argue that, to effectively manage coordination, one needs to first identify the different

types of dependencies involved, their respective characteristics and problems, and, finally, the corresponding mechanisms for dealing with these problems. Coordination theory identifies a number of basic dependencies (e.g., shared resources, producer/consumer, simultaneity, task/subtask) and suggests a few "coordination mechanisms" for each. For example, to manage dependency by shared resources (where activities share the same limited resource such as money or an actor's time), coordination theory suggests using priority orders, budgets, or biddings.

The work of Van de Van et al. [3], shares a similar standpoint also revolving their analysis around interdependencies. Van de Van et al. classify three modes in which work activities are coordinated: impersonal (i.e., following predefined procedures), personal (i.e., through one-on-one communication), and group (i.e., in planned or ad-hoc meetings). They then identify three determinants (i.e., task uncertainty, task interdependence, and organization size) and study the effects of each determinant on the three coordination modes. Their work leads to nine hypotheses that characterize the relationships between the determinants and the coordination modes.

2.1.2 The I-P-O Model and its Limitations

To varying degrees, most of the theoretical concepts on coordination (including those discussed in the previous section) are underpinned by the Input-Process-Output (or the I-P-O) model [8, 9]. As its name suggests, the I-P-O model analyzes team coordination and performance in terms of how a set of *inputs* (e.g., project requirements, schedules, completed work products) are transformed by predefined *processes* into desired *outputs* (e.g., reports, end products). Coordination theory, for instance, implies that by understanding the characteristics and risks of each task dependency (inputs) and designing some corresponding mechanisms (processes),

coordination can be achieved and desired outputs created. While such I-P-O-oriented approaches, with their coarse-grained look on coordination, are essential in team work, in recent years a number of researchers have started to agree that the I-P-O model by itself is inadequate. The main criticism of the I-P-O model is its insufficiency for characterizing and analyzing the various human, cognitive, social, and cultural factors inherent in teamwork, while such factors are often the source of complexity. Specifically, research in psychology suggests that team coordination is mediated not only by formal processes (as implied by the I-P-O model and its related theories) but by many social variables such as organizational structures, knowledge, trust, and cultural differences [9]. Failing to recognize these variables, the I-P-O model oversimplifies the complexity of teamwork. As a result, coordination strategies based on the I-P-O model may lead to teams relying only on formal processes but overlooking other significant social issues.

2.1.3 Thesis Focus on Coordination

The software development team, like many of today's organizations, strives to overcome serious coordination challenges, where large-scale, complicated, and highly uncertain and interdependent tasks need to be handled effectively and efficiently. By highlighting the limits of I-P-O-based coordination mechanisms, literature discussed in this section motivates the research questions of this study. Given that the coordination mechanisms and processes currently used in organizations and software development teams are insufficient, I set out to explore other more social and transient aspects of coordination. Specifically, I wish to better understand the concept of shared understanding and its characteristics and effects in the distributed software development team.

2.2 Empirical Studies of Software Development

The empirical studies discussed in this section point directly to the research questions by further characterizing the problem of coordination in global software development. As well, the specific issues discussed in these studies inform the choice and design of the constructs this study investigates.

2.2.1 Coordination in Software Development

A number of empirical studies have identified coordination as a critical issue in software development [10-15]. Creating a system of size and complexity involves continuous coordination among (possibly contradicting) requirements and priorities; system modules; project activities and artifacts; and technical and financial resources; as well as new and existing knowledge. All of this work, ultimately, needs to be resolved through human coordination among teams of developers and other stakeholders. Software development, therefore, can be characterized as a problem of coordination, with an emphasis on the coordination of *people* [16].

As the scale of today's software system increases, not only does the development team size increase, but the activities involved and the interdependencies among these activities also become correspondingly complex. Accordingly, the coordination needed in the development project evolves in both volume and complexity. Ineffective coordination manifests in such problems as budget overruns, delays, and poor-quality products [14] .

Despite the advance of various development processes, tools, and techniques, empirical evidence [11, 14] and the high failure rates of software projects [17] suggest that coordination is still a major challenge in software development. Based on a study of 17 large-scale software projects,

Curtis et al. [11], conclude that "coordination and communication breakdowns" are some of the most crippling challenges in software development (along with thin distribution of domain knowledge and conflicting and/or fluctuating requirements). While Curtis et al. study the "upstream" activities (i.e., requirements engineering and design), Latoza et al. [18], find similar problems during implementation. Their study finds that developers devote much effort locating the "links" between each other's work.

Section 2.1 has pointed out that coordination problems are not unique to software development but faced in the work of any large-scale organization. Certain characteristics of software development, nevertheless, may make the challenge of coordination more compounded than in other disciplines. First, the *soft* nature of software makes it exceedingly difficult to modularize a system into clear-cut components that can be distributed for development and later assembled. There is consistently a high level of interdependency in software development, among all the requirements, artifacts, activities, and, consequently, players [14, 19].

Another attribute making coordination problematic in software development is its inherent *uncertainty*. Creating software is a non-routine, "opportunistic" endeavour [20] with new knowledge constantly emerging. Rarely does development start with fully complete, accurate, and consistent knowledge about the domain, the system, and the work involved. Rather, the developers need to constantly acquire, exchange, and integrate new knowledge with their exiting knowledge base [15]. As the project progresses, developers inevitably need to modify their work in order to incorporate the newly acquired knowledge [14]. Coordination mechanisms, therefore,

not only have to resolve complex interdependencies, but also have to remain flexible in order to respond to emerging changes.

Finally, the amount of *human interaction* involved in software development exacerbates the coordination challenge. Despite conventional perceptions [21], software development is a social activity. Many studies find interpersonal communication to be a key determinant of software project outcomes [11, 20, 22, 23]. Robillard states that developers spend 59% of their time on communication activities, with 41% of these being ad-hoc, unplanned communication taking place around the water-cooler or in the corridor [20]. As much as coordinating requirements, system modules, activities, and technologies is difficult, coordinating human actors is even more so, as a multitude of human, cognitive, and social variables now need to be considered. Sections 2.3 and 2.4 will discuss these aspects in more detail.

With the goal to more effectively manage coordination, software professionals and researchers have developed many development and management processes. An examination of the major processes shows that they are, to varying degrees, "artifact-driven" [24] or, with reference to the frameworks discussed earlier, underpinned by the I-P-O model. A software process typically groups development activities into different disciplines or phases (e.g., requirements engineering, analysis and design). Within each discipline, work breakdown is defined by the identification of roles (e.g., system analysts, programmers), associated tasks (e.g., integration, unit testing), and/or artifacts (e.g., use cases, test plans). The specific execution of the tasks are described in workflow charts, timelines, or other plans of similar nature. Managers lay out schedules and assign work. Developers follow predefined procedures, produce assigned artifacts,

and use the completed artifacts as evidence that work has been done. Managers then track the artifacts with the schedules for evaluation purposes [24]. Herbsleb and Grinter [12] summarize *architectures*, *plans* and *processes* as the three main coordination mechanisms (or artifacts) in software development. Architectures provide the baseline for modularizing and distributing development work. Plans specify the major activities, schedules, and roles. Finally, processes provide detailed descriptions of how activities are to be carried out.

While these artifact-driven processes are essential, Bass [24] finds them insufficient by themselves. Processes, plans, and architectures are only effective to the extent that the future can be predicted and thus are not well-equipped to deal with the changing, emerging, and uncertain elements of software development. Architectures often become outdated as the project progresses, while plans and processes may make incorrect assumptions and estimates [12, 24].

Another problem with artifact-driven coordination is that it does not recognize or support the extensive interpersonal communication involved in software development [20]. As artifacts are the primary coordination mechanism, developers may become overly dependent on formal documents. The effectiveness of formal documentation, however, is limited [11, 12]. Curtis et al. finds "little evidence that documentation had reduced the amount of communication required" [11], noting that one particular problem is the scalability of documents, as the team grows. Other common problems with formal documentation and artifacts include tardiness, incompleteness, and mismatches in standards [11, 19]. Studies also find that developers commonly produce documents post-mortem [20] solely to comply with managerial requests. Robillard notes that,

because much communication in software development is informal, practices that explicitly support more flexible informal, interpersonal communication need to be developed [20].

Applying coordination theory to study software requirements engineering, Crowston and Krammer [7] agree that entirely I-P-O-based approaches are insufficient. Their study finds that successful coordination is achieved when the requirements engineers "just knew which features were needed, whom they had to consult for advice, whom to ask to write a specification or check a dependency, etc." Crowston and Krammer find that much of this knowledge is transient and developed, not through formal documents, but through social practices. They thus direct their study to look into the more transient and social aspects of coordination, and examine the concept of "collective mind" [25] in requirements engineering.

2.2.2 The Issue of Distance

The challenge of coordination becomes more pronounced when the project is distributed across geographical locations. The development team is now faced with the problem of *distance*. The notion of distance not only refers to geographical dispersion, but also to gaps in temporal spans; organizational structures and processes; as well as cultural values, attitudes, and norms. Many studies on global software development have discussed extensively the difficulty of coordinating across distance. Coordination breakdowns are said to most often occur at "boundaries," be they geographical, temporal, organizational, or cultural [11, 26]. Herbsleb and Grinter's study [12] demonstrates that the traditional artifact-driven mechanisms are ineffective in the distributed environment. For instance, the benefits of coordinating through the development process are reduced if the remote sites have disparate interpretations of the process.

Distance also makes it difficult to gain insight into the remote development sites. Bass [24] discusses the challenge of establishing monitoring and feedback mechanisms in the distributed project. He notes that current management approaches are incapable of closely monitoring project progress. It is often the case that developers are able to meet the specified deadlines when in fact the project is bombarded by a multitude of problems that are not identified by the artifact-driven approaches.

Carmel and Agarwal [27] claim that distance also exacerbates coordination problems indirectly through its effects on communication. There is a substantial body of literature, particularly in the field of Computer-Supported Collaborative Work (CSCW), on the effects of distance on communication [12, 27, 28]. Most obviously, distance reduces communication availability and frequency [28]. Informal, ad-hoc communication, in particular, suffers from geographical dispersion. Despite advances in CSCW technologies, the loss of context is unbridgeable. Also people usually give lower priority to messages from strangers at the remotes sites [12].

Another much-cited difficulty in the distributed project is the challenge of the "transfer of best practices." Herbsleb and Grinter [12] note that the distributed development sites often adhere to different processes and practices. As well, the activities and/or standards at one site may be incomprehensible to other sites, creating significant overhead when work needs to be integrated. Although there are often explicit attempts by the organization to transfer best practices across sites, the outcome is usually unsatisfactory [26]. Cusumano [29] surveys software organizations in North America, Europe, and Asia and finds significant divergence in the development practices used in these regions. For example, while rigorous design reviews are the norm in

Japan (and less so in North America), pair programming is almost non-existent in Japanese software organizations but commonplace in other regions.

2.2.3 The Issue of Culture

Geographical dispersion introduces another challenge - managing cultural diversity in the team. In the distributed software project, developers of diverse cultural backgrounds are brought to work together, each bringing a unique set of values; beliefs; behavioural norms; and communication and problem-solving approaches. Some researchers and professionals alike have started to recognize the impact of intercultural differences on the distributed development team [13, 27, 30-38]. Carmel even claims culture to be "the most perplexing factor" in the distributed project [30].

Intercultural differences affect software development in many different ways. The most notable manifestation may lie in the communication activities. Cultural groups differ in their communication processes in aspects such as preferred communication forms (e.g., written or oral), the degree of explicitness, and other customs and norms (e.g., who initiates communication and how). Krishna [35] finds that Indian developers deliberately adjust their communication styles between Japanese and American clients. Cultural differences may cause unexpected misunderstandings in communication. For instance, Borchers [39] notes that acknowledgement from Asian developers is often misinterpreted as agreement by their North American counterparts.

Team structure is another area where intercultural differences may wreak havoc. Different cultures not only have differing concepts about the team, but they may consequently differ in

ways to structure and manage the team. Studying a software team composed of Indian and Jamaican developers, Walsham [40] cites differences in team management as one of the main sources for conflicts. While the Indian developers find the Jamaicans to be too laid-back and to have seemingly little respect for management, the Jamaican developers are frustrated with the Indians' more autocratic, "adult-child" mentality.

Developers' understanding of the technical system may also bear a cultural bias. Borchers [39] reports that system requirements may be interpreted differently by developers of different cultures. He also observes that design approaches, particularly in how abstractions are chosen, differ along cultural boundaries. While Borchers does not do an in-depth study of the cultural impact on design, he finds such differences to cause overhead in the integration phase.

Cultural bias also affects development practices. Studying requirements engineering in a Thai software organization, Hanisch [41] finds the process to be heavily influenced by the hierarchical and collectivist orientation of the Thai culture. The respect for hierarchy in the Thai society adds overhead and delay to the project, as decisions need to be reviewed and approved at different layers in the organization. Hanisch also finds that Thai developers rarely reject or criticize client inputs, as they are reluctant to disturb the "harmony". While Hanisch studies a homogeneous Thai team, one may expect conflicts and misunderstandings if the team were to include remote developers from other cultures.

2.2.4 Thesis Focus on Empirical Software Engineering

The choice and design of constructs that this study investigates are heavily influenced by the studies discussed in this section. In particular, these studies underline areas in software

development where coordination is likely to be problematic. Herbsleb and Grinter's study [12] highlights the challenge of establishing shared *processes and practices* among the distributed sites. The work of [7, 11, 23, 32, 33, 39, 41, 42] supports that *requirements engineering* is an area where problems commonly arise. Borchers's work [39], though more anecdotal than empirical, suggests culturally different approaches to *architecture and implementation.* Bass [24] and Robillard [20] highlight communication problems and the difficulty of getting immediate and accurate feedback in the distributed team. I extend the findings of these studies to define the constructs.

2.3 Shared Cognition

Recognizing the insufficiencies of I-P-O-based coordination, researchers in cognitive sciences and psychology have started to examine the softer aspects of coordination — concepts such as shared mental models and collective mind, among others [8, 9, 25, 43]. Though differing in their theorizing, the key idea of these concepts is that team coordination is improved when the members establish a common framework for their knowledge, approaches, identity, expectations, actions, values, and beliefs. The benefits of shared cognition in the team lie in its *explanatory*, *predictive*, and *adaptive* functions. First, as the team members develop a common framework, they can better understand and explain the task, the environment, and the actions of others. Second, the members can draw from this shared knowledge to make predictions about the team (e.g., what other members will do, what effects their actions will have, what resources they will need). Finally, and perhaps most importantly, based on the predictions, the members can more efficiently devise new strategies and adjustments to coordinate with the others and to respond to emerging events.

2.3.1 Shared Mental Models

Shared mental models are defined as organized knowledge structures that team members hold about issues such as the "tasks, each other, goals, and strategies" [43, 44]. Shared mental models allow team members to draw on their common, well-organized knowledge as the basis for selecting actions that are coordinated and consistent with the rest of the team [45]. Klimoski and Mohammed [43] argue that an effective, well-coordinated team will likely hold multiple models to reflect different problem domains. At the most rudimentary, the team would hold a work-related model and a team-related model. The work-related model further includes the technology/equipment model and the job/task model. The technology/equipment model contains knowledge about the team's use of technology, operating procedures, and likely problems. The job/task model describes how tasks are accomplished in terms of procedures, strategies, and likely contingencies. The team-related model, on the other hand, includes the interaction model and the team model. The interaction model holds information about the members' roles and responsibilities; communication channels; interaction patterns; and the flow of information within the team. Finally, the team model refers to specific knowledge about the individual members' attitudes, skills, and preferences [45].

Cannon-Bower and Salas [8] offer a slightly different categorization. They argue that the knowledge shared in the team falls into one of four categories: task-specific knowledge; task-related knowledge; team members' knowledge of one another; and shared attitudes and beliefs. Task-related knowledge refers to the procedures, sequences, and strategies necessary to perform a particular task. Task-related knowledge is not specific to any one task but holds across a variety of (albeit similar) processes. Knowledge of team members describes understanding of

team mates' preferences, strengths, weaknesses, and tendencies (corresponding to the Mathieu et al. team model). Finally, shared beliefs and attitudes hold that team members have compatible attitudes and approaches about the task and the environment, when they have similar values, motivations, and goals.

2.3.2 Collective Mind

A related concept is collective mind theory, proposed by Weick and Robert [25]. The theory explains group performance in situations requiring exact, continuous operational reliability. Weick and Robert's work is based on studying flight deck operations on aircraft carriers and investigates the question: "W*hat makes the operations on the flight deck, a place with 'a million accidents waiting to happen,'[46] so reliable?"*

Weick and Robert assert the answer is that members of an organization concerned with reliability are capable of enacting more fully developed mental processes — conceptualized as "collective mind." Such mental processes enable the members to acquire a better understanding of the complexity they face and to respond with fewer errors and greater efficiency. Collective mind differs from the concept of shared mental models in that it focuses on the *practices* that transfer understanding (rather than the contents of the understanding). Weick and Robert identify three practices typifying collective mind: contribution (i.e., an individual constructing his/her own actions), representation (i.e., an individual envisaging a personal mental model of the group's actions, needs, and structures), and subordination (i.e., an individual aligning his/her actions to the group as envisaged). Weick and Robert argue that collective mind is achieved when the team adheres to these practices. They also imply that, by studying these practices, one can better analyze team coordination and performance.

2.3.3 Application in Software Development

A number of researchers have applied the concepts of shared cognition to study software development [7, 18, 47, 48]. Espinosa et al. [47] investigate the effects of shared mental models in distributed software development. Through interviews, surveys, and archival studies, they find that shared mental models have positive effects on team performance, and that compared to the collocated development team, shared mental models provide greater value in the distributed environment. Levesque [48] studies the effects of time and role differentiation on shared mental models. They find that, contrary to common perceptions, developers do not converge in their mental models if they do not expect to collaborate in the future.

Crowston and Krammer [7] study collective mind in the requirements engineering team, characterizing coordination breakdowns in terms of contribution, representation, and subordination. With regard to contribution, Crowston and Krammer found that most of the effort in the team is directed towards accomplishing technical tasks rather than explicitly promoting shared understanding in the team. In terms of representation, the developers often have confused or contradictory interpretations as how and why certain requirements are defined. Breakdowns in representation then lead to discrepancies in subordination. Individual developers all work in the way that they consider the most positive to the team; however, failing to see "what the other sees" [7], the team lacks an overall goal, causing delays and frictions.

2.3.4 Thesis Focus on Shared Cognition

Literature on shared cognition contributes both to the study's research questions and to the construct design and choice. There is increasing emphasis in the software engineering

community on the human and social factors of software development [49]. Concepts such as shared mental models and the collective mind provide a theoretical standpoint from which one can study these human, social phenomena. Crowston and Krammer [7] explain how the concepts of collective mind and shared mental models are particularly suitable for studying software development. The concepts are traditionally used to analyze *high-performance* organizations that perform *non-routine work*, under *changing, complex task conditions* [7, 45]. Crowston and Krammer argue that these factors are also present in today's software development team. Furthermore, the distributed development team is faced with another dimension of complexity — limited communication opportunities as a result of physical dispersion.

This body of literature, in particular the categorization of shared mental models, also informs my choice and design of the study constructs. Specifically, the study investigates how cultural diversity affects the team's knowledge that is task-specific, task-related, and/or related to the team's goals and motivations.

2.4 Culture

This study also arises out of an interest in the effects of intercultural differences on the coordination. I therefore draw from a number of classical cultural studies to shape the research questions and to design the study constructs.

2.4.1 Concept of Culture

Depending on the field of exploration, there are differing definitions of culture [50-55]. For the goal of this study, an appropriate definition is supplied by Spencer-Oatey, who defines culture as "a fuzzy set of attitudes, beliefs, behavioural norms, and basic assumptions and values that are

shared by a group of people, and that influence each member's behaviour and her/his interpretations of the 'meaning' of other people's behaviour" [55]. This definition makes a number of implications that relevant for this study.

First, the definition suggests that culture is *shared* by a group and, at the same time, *distinctive* from other groups. The values and practices adhered to by one culture are not necessarily understood or agreed to by others. In intercultural interactions, these differences may lead to misunderstandings and other conflicts.

Second, culture is *layered.* At the most rudimentary, culture consists of two levels: an invisible values level and, on top of that, a visible behaviours/artifacts level. The values level contains ideas and beliefs about issues such as the sense of time, the sense of space, the relation of people, and the function of rules and laws. These values are invisible but manifest through the behaviours and artifacts (e.g., rituals, norms, social institutions, clothing, and fine arts). While differences in behaviours and artifacts can be readily identified, differences in values must be inferred and, therefore, often lead to misunderstandings and confusions in intercultural interactions [56].

Finally, the definition suggests that culture is both an *influencing* factor on behaviours and an *interpretive* factor of behaviour. In other words, the function of culture is not only in guiding one to form his/her actions, but also in how one interprets others' actions. This interpretive role of culture is especially important when studying intercultural interactions [56].

2.4.2 Cultural Frameworks

Researchers have devised a number of frameworks that analyze intercultural differences in values and behaviours. While this study does not use any specific models, these frameworks contribute to the design of the study constructs by highlighting areas where intercultural differences are likely to affect the performance of the software development team. The following sections summarize some models commonly quoted in intercultural research.

2.4.2.1 Hofstede

Surveying IBM employees in over forty countries, Hofstede [51, 57] develops a set of cultural indices. The indices are relativistic scales for a culture's values and approaches regarding five issues: power distance, individualism/collectivism, masculinity/femininity, uncertainty avoidance and long-term/short-term orientation [36].

The power distance index measures the extent to which a culture embraces social inequality. In a culture with high power distance there exists an established hierarchy of power, based on status, wealth, intellectual capacity, or some other factors. A culture with low power distance, on the other hand, considers every individual as equal, despite differences in power, status, or wealth.

The individualism/collectivism index describes how an individual is perceived: as an independent entity or as part of a larger, cohesive group. An individualistic culture is one where individual interests take precedence over collective ones; everyone is expected to look after himself/herself. A collectivist culture is one where people are integrated into strong, cohesive groups; people give allegiance to the groups to which they belong.

The masculinity/femininity index describes the distribution of roles between genders. A masculine culture has distinct, fixed gender roles. Men are expected to be assertive, strong, and focused on material success while women are gentle, caring, and concerned with the quality of life. Gender roles in a more feminine culture are more fluid. Both men are women can be tender and focused on improving the quality of life.

The uncertainty avoidance index indicates the tolerance a culture has for unfamiliar or ambiguous situations. A culture with a high ranking in this dimension relies upon strict, specific rules and procedures to mitigate uncertainty. A culture with low uncertainty avoidance is more comfortable handling unknown events and thus relies less upon rules.

The long-term/short-term orientation addresses a culture's approaches to need gratification. A culture with long-term orientation prescribes to long-term commitments and perseverance towards slow results. A culture with short-term orientation is pragmatic, welcoming to changes, and looking for rapid compensations.

2.4.2.2 Hall

Hall [50] identifies two dimensions of culture: communication patterns and sense of time. First, Hall distinguishes cultural groups based on the amount of information that needs to be directly, explicitly stated for communication to be effective. A high-context culture is one where most information is "pre-programmed" into the external setting (e.g., time and location of communication) or the parties involved (e.g., relationships between the sender and the receiver; gestures; silences). The actual message that is transmitted contains little information. It is the receiver's responsibility to interpret the message, based on an awareness of the contextual cues.

Conversely, a low-context culture is one where information is transmitted through explicit, direct, unambiguous messages. The sender is responsible for ensuring that the message can be readily interpreted without confusion.

Hall also examines how different cultural groups perceive and structure time — monochronically or polychronically. A monochronic culture views time in a sequential fashion. One task is completed before another is started, following a precise and detailed schedule. Interruptions are uncommon and there is a clear distinction between personal and work time. In a polychronic culture, time is much more flexible. Multiple tasks can be handled simultaneously. Interruptions are common and time is subordinate to personal relationships [56].

2.4.2.3 Trompenaars and Hampden-Turner

Examining the impact of culture on business and management processes, Trompenaars and Hampden-Turner [58] identify seven value dimensions: universalism/particularism, neutral/emotional, specificity/diffuseness, achievement/ascription, communitarianism/individualism, human-time relationship, human-nature relationship, and internal/external.

The universalism/particularism dimension describes a culture's preference for rules or relationships. A universalist culture relies on established rules under all circumstances. A particularist culture focuses on the present circumstance and is inclined to give special consideration based on the uniqueness of the situation or relationship.

The neutral/emotional dimension addresses how emotions are expressed in a culture. Members of a neutral culture are reluctant to reveal their feelings and thoughts in public while people in a emotional culture are more direct in their expressions of feelings and thoughts.

The specificity/diffuseness dimension measures the degree of involvement. In a specific culture, each (business) relationship is precisely defined around its limited context. Co-workers are less likely to establish relationships outside of the work context. In a diffuse culture, different areas of life are interconnected, with arenas like work and family life intermingling.

The achievement/ascription dimension describes how status is accorded in a culture. An achievement-oriented culture accords status by performance and material success. An ascriptive culture derives status from birth, kinship, gender, and seniority.

The communitarianism/individualism dimension, virtually identical to Hofstede's collectivism/individualism index, describes a culture's focus on the group or the individual.

The human-time relationship is closely related to Hall's discussion of polychronic/monochronic time perceptions, measuring how a culture structures time.

Finally, the internal/external dimension addresses a culture's attitudes towards the natural environment. An internalistic culture believes that control lies within the individual, while an externalistic culture fears and subjugates nature.

2.4.3 Other Research and Criticisms

The frameworks discussed in the previous sections have been widely quoted and applied in cultural research. They have also gained popularity in the fields of business management and organizational science, as they provide easy-to-understand and systematic approaches for analyzing intercultural interactions and conflicts [56]. Laroche [59], leaning on the work of Hofstede and Hall, studies intercultural collaboration in the technical team. He identifies a number of areas where intercultural differences are likely to precipitate conflicts. For instance, the manager-employee relationship may be rocky if the manager and the employee differ significantly in power distance.

In recent years, however, some questions have been raised regarding the robustness and applicability of the classic cultural frameworks [38, 56, 60]. One criticism is that the value dimensions depicted by these frameworks are the result of limited data, uncovered by a limited number of questions. The lack of substantial, solid empirical data threatens to hide other important dimensions or lead to inaccurate causality [60]. Others charge that the frameworks assume "cultural homogeneity" [38] and fail to consider influences such as professional, organizational, gender, regional, and generational differences that exist *within* any one culture. As pointed out by Walsham [38], the models take a rather deterministic stance and do not consider the dynamic nature of culture (i.e., humans are capable of making adjustments to deal with external, including cultural, influences). Other critiques argue that the models are outdated and no longer applicable [56].

2.4.4 Thesis Focus on Cultural Research

The cultural research discussed in this section contributes directly to the research questions and to the study constructs. The work of Hofstede, Trompenaars and Hampden-Turner, and more recently, Laroche illustrates that intercultural differences in values and behavioural norms are often the cause of conflicts and other challenges in today's culturally diverse organization. This study thus investigates the effects of culture on coordination in the global software development team – where developers of diverse cultural backgrounds continuously and closely collaborate, often through virtual channels.

The specific cultural models and patterns provide the study with a set of readymade constructs. Although the criticisms on these frameworks are valid, I consider the frameworks nevertheless valuable for providing a starting point to examine intercultural interactions. To avoid the pitfalls of over-reliance on deterministic and possibly outdated value orientations, the study will take into account intervening constructs that consider other contextual, non-cultural influences. To keep the study tightly focused, I draw from Laroche's work [59] and identify seven dimensions that may have grater influence on the distributed development team. The selected dimensions are uncertainty avoidance; power distance; collectivism/individualism; sense of time; communication patterns; and expression of emotions. Section 3.3 discusses the constructs in more detail.

2.5 Qualitative Research Methods

In the past decade, there has been an increasing interest in studying software development empirically [61]. Researchers study issues related to software development processes, techniques, and products through such methods as experimentations, surveys, archival analysis,

and simulations [62]. Many of the studies are completely quantitative in that they use statistical and/or mathematical methods for both data collection and analysis. Others may include a small portion of qualitative research, the goal of which is to attain a general understanding about the problem domain, before starting more extensive investigation with quantitative methods. To date, few studies have systematically and extensively made use of qualitative data to study and explain software development. Yet, as Seaman [49] argues, software development is a social activity in which a magnitude of human, organizational, and cultural factors is the root of the problem and requires close examination. She argues that quantitative studies alone "have not delivered" [63], as they are not suitable for studying the "people problems" [49]. Others have started to agree that empirical studies of software development need to incorporate more rigorous qualitative methods (as evidenced by [64]).

I choose a qualitative research strategy for two reasons. First, the subject of interest — shared cognition and culture in the distributed development team — can not be adequately studied through entirely quantitative methods. Second, there has been little previous work done in this area. Conducting large-scale, quantitative surveys or experiments is unlikely to yield any meaningful results. For these reasons, I adopt a qualitative, exploratory case study strategy.

In this work, I consult a number of sources in designing the research methods. Yin's [65] work on case study research informs the overall study design, as well as the strategies and techniques for addressing validity issues. Brewer [66], Seaman [49], and Kitchenham [61] present the definitions of and methods for interviewing as a data collection method. I follow Miles and Huberman's advice [67] to define the a priori constructs (i.e., specific issues to be investigated,

as suggested by the research questions). Data analysis is guided by Eisenhardt's constant comparison approach [68], which in turn is based on the grounded theory [69, 70]. The detailed coding and memo-writing techniques are informed by the work of Miles and Huberman [67] and Coffey and Atkinson [71]. I am also guided by a number of exemplar empirical studies, both in software engineering [63, 72] and in other disciplines such as organizational sciences [73].

The specific contributions of these methods and studies are described in detail later, when the study's research design is presented, in Chapter 3.

CHAPTER 3
METHODOLOGY

3.1 Overview

The study described in this thesis explores the effects of cultural diversity on the distributed team's shared understanding regarding issues such as development processes, system requirements, and scheduling. The study design is what Yin terms "exploratory" "embedded single case" design [65]. The study is exploratory because there are currently no well-formulated theories or hypotheses on the topic. The goal of the study is not to confirm or dispute existing theories but to generate a set of propositions that characterize the problem and that may be used to further future research.

Embedded design denotes multiple levels of analysis (as opposed to holistic design, which has a single unit of analysis) [65]. I conduct the investigation at two levels. First is the *team* level, which examines shared understanding in the team as a whole. I define questions that explore specific incidents and conflicts in the team, as well as the overall project coordination and performance. The other analysis unit is the *individual developer*. Questions are designed to study the developer's personal perspectives and experiences in the project. The embedded design allows more extensive analysis and keeps the study focused on the research questions [65, 74].

A single-case design is chosen for both practical and theoretical reasons. The chosen case is considered "revelatory," [65] meeting Yin's rationale for the single-case design. As discussed in more detail in Section 3.3, the case is the development of an information system, involving a

Canadian prime and two Italian subcontractors. Characteristics of the case (e.g., physical dispersion, cultural diversity, subcontracting arrangements among multiple organizations) are common of distributed software development projects. There have been no previous studies on this subject (i.e., shared understanding in the distributed team and the effects of culture).

The specific steps and techniques of the research method are described below, along with studies in the literature that inform my design.

3.2 Other Cases

This study is one part of a multi-pronged research effort investigating the effects of culture on global software development. Besides the Canadian-Italian subcontracting case described in this thesis, another investigator, Eve MacGregor, and I have examined four other cases:

- Case A: a Canadian company that used to outsource to India but recently started its own offshore development centre in China;
- Case B: a Nepalese software house with clients in USA and Europe;
- Case C: a US company outsourcing to Russia;
- Case D: an outsourcing relationship between Canada and India.

For these four cases, we have thus far conducted a total of 12 interviews with 7 subjects. Table 3-1 summarizes information about these cases, the companies involved, and the numbers of subjects and interviews for each case.

The choice to focus on only one case (i.e., the Canadian-Italian subcontract) in this study is mainly due to resource and scope limitations. I am unable to conduct detailed cross-case analysis

given the time constraint. I therefore decide to focus on the Canadian-Italian case which, from the data collected, suggests significant gaps in the distributed sites' shared understanding.

The case described in this thesis informs the rest of the research effort. It is nevertheless inappropriate to consider this case and the others together as a multiple-case study, as there is no clear replication logic among the cases. Still the cases, when examined together may yield findings that corroborate or theoretically replicate one another. Future research efforts may also validate the findings from this study with other cases.

Table 3-1 Summary of other Cases

Case	Description	# of Subjects	# of Interviews
A	Case A focuses on a Canadian company specializing in Information Management Systems. The company used to outsource development and testing functions to a software consulting company based in India. This outsourcing relationship was terminated in the summer of 2006. The Canadian company has since then started its own offshore centre in China.	6	9
B	Case B focuses on a contract software development company located in Nepal. The company has clients located in the USA and Europe.	1	1
C	Case C focuses on a outsourcing relationship between a USA-based company specializing in security technologies and a Russian software consulting company.	1	1
D	Case D examines on a software company with the headquarter in the US and a major development centre in Canada. The Canadian site further outsources to an Indian consulting company. The case focuses on the interactions between the Canadian and the Indian site.	1	1

3.3 Case Description

The case that provides the subjects for this study is a subcontracting relationship between a Canadian information systems company (hereafter referred to as CanTech) and two of its

subcontractors in Italy (hereafter referred to as I-Sys and SofTar, respectively). The project under study is the development of an information system. The system consists of three critical components. CanTech subcontracts one of the components (hereafter referred to as the SBS subsystem) to I-Sys, an information solutions provider in Rome. The subcontract is worth an estimated CAN $80 million. I-Sys is responsible for both hardware and software development of SBS. I-Sys, in turn, subcontracts part of the software development to SofTar, a software supplier in southern Italy.

The project starts in December 1999. I-Sys initially estimates to deliver the SBS subsystem within six months. It is May 2004, however, when the bulk of the development is completed and delivered to CanTech. As of this writing, there is ongoing integration of SBS with the rest of the system. The overall information system is scheduled to be deployed in the summer of 2007.

During the project, between 20 and 30 developers from CanTech work on the subcontract, while I-Sys and SofTar, together, provide over 100 developers. On the software-specific components, two and a half developers from CanTech are responsible (two full-time and one part-time). It is unclear how many human resources from the two Italian organizations are assigned to the software-specific part of the project.

Prior to the project, I-Sys, the primary subcontractor, presents to CanTech a process document, describing the development process that will be used in the project. The process document is based on a set of international software engineering standards (hereafter referred to as the ISE

standards). The ISE standards are selected because they conform to CanTech's internal development processes. SofTar has its own process document, also based on the ISE standards.

Communication between CanTech and the subcontractor sites relies on three channels. First, I-Sys and SofTar regularly submit process-required plans and reports to CanTech for review and feedback. Second, teleconferences and videoconferences are set up. The teleconferences/videoconferences are used for two purposes. A weekly *status meeting* is arranged for managerial discussions (e.g., resource distribution, scheduling issues). The developers, on the other hand, arrange *working group meetings* for technical discussions. Unlike the status meetings, the working group meetings are not held regularly but are called as needed. Finally, the CanTech developers travel regularly to the Italian sites for face-to-face meetings and reviews.

Two and a half years into the project, CanTech demands a Critical Design Review (CDR), where I-Sys presents its final system design. As a result of the discoveries made at the CDR, CanTech realizes that I-Sys and SofTar are doing little to follow the promised development processes. CanTech also finds the system quality and project progress unsatisfactory. To respond, CanTech stations a number of developers at I-Sys and SofTar. These developers assist I-Sys and SofTar in their development activities, while reporting back to CanTech on the project status. Other CanTech developers, meanwhile, start visiting the Italian sites more frequently and work with the Italian developers on requirements verification, reviews, and testing. CanTech also starts arranging software audits and walkthroughs more frequently at the Italian sites to better monitor the project progress.

A major conflict in the project occurs after a software audit. CanTech requests the audit and sends three developers to SofTar for the purpose. I-Sys and SofTar, seemingly unfamiliar with the audit purpose and procedures, present the CanTech developers with boxes of hard copies of codes to review. After the audit, frustrated and tired, the CanTech developers express their dissatisfaction with the system quality in an internal memo. The memo, for reasons unclear, is sent to I-Sys. The managers and developers at I-Sys are outraged by the criticisms expressed in the memo. CanTech subsequently issues a formal apology.

3.4 Constructs

For the study I design an a priori specification of constructs. The constructs represent the key issues the study investigates and are derived from the literature. Defining a priori constructs is not a common practice in exploratory studies. I nevertheless adopt this strategy here to better focus the research questions and to guide data collection and analysis [68].

3.4.1 Constructs: Shared Understanding

The predefined constructs fall into two categories. The first reflects various aspects of shared understanding. I first draw from literature on shared cognition to identify knowledge areas that need to be shared within the team. I then consult studies of software development to specify seven areas where coordination problems are likely to occur — and where shared understanding may have the greatest benefits.

Process and Practices captures issues reflecting the team's shared knowledge about the development processes and practices. *Requirements* describes if and how the distributed

developers have shared knowledge and approaches about the systems requirements. *Technical Details* refers to the team's knowledge about the technical activities (e.g., design, implementation, testing) carried out at the remote sites. *Scheduling and Project Status* reflects the extent to which the developers understand the progress, challenges, and plans at the remote sites. *Resource Management* describes the developers' shared conceptions about the location, availability, and flow of expertise in the team. *Shared Goals* reflects whether the distributed sites are alike in their goals, motivations, and beliefs. Lastly, other *Task-/Team-Related Issues* refers to the team's knowledge, attitudes, and approaches regarding any other issues that are not specific to particular tasks but that affect the overall team performance. Examples of such issues include communication mechanisms, conflict-resolution approaches, and time-management skills. Table 3.1 includes sample questions used to investigate each of the constructs.

3.4.2 Constructs: Culture

The second category of constructs covers selected cultural dimensions. I draw from the cultural frameworks discussed in Chapter 2, and, consulting studies of software development and Laroche's work [59], identify six constructs representing those value/behavioural orientations likely to have affect the development team. *Uncertainty Avoidance*, *Power Distance*, and *Collectivism/Individualism* are derived from Hofstede's model. *Sense of Time* and *Communication Patterns* are based on Hall's discussion of time and communication contexts. Finally, *Expression of Emotions* is drawn from Trompenaars and Hampden-Turner's cultural index.

Table 3-2 Sample Questions for Category 1 Constructs

Construct	Related Questions
Process and Practices	▪ Do the distributed sites follow the same process? ▪ Do the distributed sites follow the same practices? ▪ If there are differences, are they intentional? Are the distributed sites aware of the differences? ▪ What effects do the differences have? Positive or negative?
Requirements	▪ What is the requirements engineering process? ▪ Are there any misinterpretations of the requirements among the distributed sites?
Technical Details	▪ How are design/implementation/testing activities distributed among and carried out at the distributed sites? ▪ Do the distributed sites collaborate on design, implementation, and testing activities? How? ▪ How much insight do you have into the technical activities at the remote sites?
Scheduling and Project Status	▪ How long does it take for you to discover the problems encountered at the remote sites? ▪ How are schedules determined?
Resource Management	▪ Do the distributed sites collaborate on resource management issues? How? ▪ How much insight do you have into the resource management and team structure at the remote sites? ▪ How long does it take for requests for resources to be answered by the remote sites?
Goals	▪ Do you observe differences in goals, motivation, and commitment among the distributed sites?
Task-/Team-Related Issues	▪ Do you use any special communication and/or collaboration tools? ▪ How do you draw the boundary of the "team"? ▪ Do you observe any communication mishaps? ▪ How are problems reported and handled? ▪ Do you observe any specific conflicts?

3.4.3 Emerging Constructs

The construct list is kept open. The defined constructs are continuously redefined, modified, or even removed, if necessary, to reflect the emerging data. New constructs are created to capture unforeseen concepts and themes. Some of the emergent constructs may augment the existing two categories and describe issues related to shared understanding and/or culture. Others may be considered *intervening* constructs not directly related to the concept of culture. The intervening constructs are not the primary concern of the study but nevertheless emerge as bearing significance on shared understanding in the global software development team. The data suggests two such intervening constructs: *Nature of the Relationship* between the distributed

sites and *Communication Mechanisms*. They will be discussed in greater detail, along with the related findings, in Chapter 4.

3.5 Data Collection

Data is collected between December 2005 and September 2006. During this period another investigator, Eve MacGregor, and I conduct semi-structured interviews [66] with six subjects. All six subjects are developers from CanTech who have worked on the project. Nine interviews conducted as follow-up interviews are held with three of the subjects. Due to limited resources we are not able to interview any developers from I-Sys and SofTar. This omission may affect the study's reliability and will be addressed in Section 3.6. The interviews range from 30 minutes to approximately one hour. The roles and responsibilities of the subjects interviewed are described in Table 3.2.

Table 3-3 Subjects Description

Subject	Description of Subject
Blake	▪ Lead engineer for the overall information system ▪ Visits the Italian sites periodically
Colin	▪ Lead software engineer for the overall information system ▪ Initially visits the Italian sites periodically ▪ Towards the end of the project, stays in Italy for extended period of time ▪ In total spends more time in Italy than any other Canadian developer
Jack	▪ Software engineer ▪ Visits the Italian sites periodically
Holmer	▪ Resident interface at I-Sys between mid-2001 to mid-2003 ▪ Responsible for (1) monitoring progress at I-Sys, (2) reviewing technical documentation produced by I-Sys, and (3) reporting back to Can-Tech
Stephen	▪ Two assignments in Italy ▪ First at SofTar for three months; initially sent to monitor software testing but becomes involved in monitoring and reviewing design work because of delays in the project ▪ Second at I-Sys for 13 months; initially scheduled for a three-month assignment
Howard	▪ Resident at SofTar for 9 months ▪ Initially scheduled for a two-month assignment assisting software testing ▪ Becomes involved in requirements verification and testing because of delays in the project

The interviews are semi-structured because each interview is guided by a predefined interview protocol. The protocol includes a specific set of questions, eliciting answers that are the objective of the interview. However, most questions are open-ended and intended as cues to memory, as well as for soliciting information not foreseen by the researchers. Throughout data collection, we continuously revise the questions, as we become more knowledgeable about the nature of the case.

All the interviews are audio-taped in their entirety. Eve and I then transcribed the interviews verbatim. Although Cox [75] recommends the practice of transcribing and analyzing an interview immediately after it is done, due to limited resources we are not always able to do so. For any interview not immediately transcribed, we review the audiotape and make extensive field notes before the next interview. Based on the analysis results (or field notes) we then determine whether a follow-up interview with the subject is necessary. The follow-up interview is conducted when there are issues that need further exploration or when the first interview does not cover all the topics of interest. A sample interview protocol can be found in Appendix A.

3.6 Data Analysis

The raw data collected for this study is analyzed using the "constant comparison method" promoted in the grounded theory [69, 70] and further elaborated by Eisenhardt [68] and Seaman [63]. The goal of the analysis is to (1) find empirical evidence for the predefined constructs, (2) identify emergent (intervening) constructs, and (3) identify possible relationships among the constructs.

The analysis is carried out in an iterative process consisting three stages: *coding*, *memo-writing*, and *comparison*.

3.6.1 Coding

Coding is the activity of using labels or tags to classify and assign meaning to the pieces of data collected in the study [67, 69]. The labels and tags form the *codes* and represent issues of interest in the study. In this work, most of the codes can be organized into a hierarchical structure (as shown in Appendix A.3). The higher-level codes are keyed to the constructs, while the lower-level codes denote related issues and/or incidents. As analysis progresses, I also create new codes to denote emerging concepts and patterns.

The purpose of coding is twofold. First, coding helps to "efficiently retrieve and organize" large amounts of data [71]. By assigning codes to data of varying sizes – words, phrases, or paragraphs – data is effectively organized into categories. Rather than surveying the entire data set, I can retrieve any specific piece of data by selecting the related code(s) [67]. Second, through the process of coding, I become "intimately familiar" [68] with the data.

I use Qualrus, a qualitative analysis software package, to code the data. Each interview transcript is examined and coded. Coding is an iterative process. Each of the coded transcripts is reviewed several times (both as a standalone data source and in conjunction with other transcripts) to ensure that the codes are assigned consistently and that all the key patterns and issues are captured by one or more codes.

A list of all codes with frequencies can be found in Appendix A.2. Samples of coded segments can be found in Appendix A.4.

3.6.2 Memo-writing

As coding progresses, I write field notes to capture any impressions, themes, pointers, and questions that emerge from the data. After coding is completed for an interview, I examine in detail the codes, coded text segments, and any field notes written. I then write a summarizing memo on the key findings from the interview. The memo focuses on each individual aspect of shared understanding, the related cultural factors and their effects, as well as any other issues or phenomena that appear to be of significance. Through writing the memos, I document any emerging, potential relationships among the constructs. A sample memo can be found in Appendix A.5.

3.6.3 Comparison

For the first two interviews reviewed and coded, I examine and compare the memos for these interviews, to determine any similarities and contradictions between their findings. I then list, in the form of propositions, any conclusions that can be formulated *if these two interviews were the only two in the data set.* The comparison, similar to the memos, focuses on the various aspects of shared understanding, relevant (cultural) factors, and any possible relationships.

After the first two interviews are analyzed this way, I code the third interview and write the corresponding memo. I then determine whether the third interview supports or refutes any of the previously formulated conclusions. If the new findings support a conclusion, confidence in the conclusion is strengthened. If the findings from the third interview refute a conclusion, I review

all three interviews (along with the codes, field notes, and memos) to search for evidence explaining why the findings diverge. I then revise the conclusions to incorporate the new evidence. If the third interview suggests any additional conclusions, they are added to the list of propositions. I repeat the process with each subsequent interview until I have used the entire data set.

The motivation for using the constant comparison method is both practical and theoretical. First, the volume of data is simply staggering. If it were to be analyzed as a whole, it would be difficult to find any meaningful themes and concepts. By limiting the focus to only two interviews at a time, it is easier to focus the analysis on the research questions and to identify important patterns and relationships. The second motivation is to provide solid empirical grounding for the findings. By checking the emergent conclusions with each individual data source (interview), I ensure that the resultant conclusions are supported by the entire data set.

3.7 Validity Concerns

According Yin [65], any empirical case study must take into account four types of validity concerns: construct validity, internal validity, external validity, and reliability. Each of these validity concerns is associated with a number of issues that may impact the soundness of the study. The following sections discuss these issues and any techniques used to address them in this study, along with data accuracy, a subject related to all four types of validity [63].

3.7.1 Construct Validity

Construct validity is concerned with establishing appropriate operational constructs and measures for the concepts the study investigates [65]. The issue should be discussed at two

levels. The higher-level question is whether the study constructs, collectively, represent the concepts of interest [63] – in this case shared understanding and culture. Given the complexity, I recognize that the collection of constructs used in this study is inadequate to capture the full richness of the concepts (e.g., the collection of the seven selected cultural indices certainly does not represent the full dimension of culture). I nevertheless argue that the selected constructs make a reasonable compromise between practical (e.g., availability of data, time limits) and validity concerns. The argument is supported by the literature. As discussed in Chapter 2, the constructs are selected based on empirical studies of software engineering and cultural research to represent the areas considered to be the most problematic.

At the lower level, construct validity has to do with demonstrating that the study accurately measures and analyzes the selected constructs. For this purpose, Yin [65] suggests the practices of maintaining a "chain of evidence" between the research questions and the findings. In this study, the chain of evidence can be observed from the research questions to the constructs, the interview questions, the data, and, finally, to the findings. Sample interview protocols, codes, coded data segments, field notes, and memos (documenting how I arrive at the findings) are presented in Appendix A. In Chapter 4, I discuss the findings along with supporting evidence from the data, while explaining how the findings answer the initial research questions.

3.7.2 Internal Validity

Internal validity, addressing the validity of any *causal* relationships drawn by the study, is normally not applicable to exploratory studies. As this study is exploratory and does not report any causal inferences, I make few attempts to mitigate the related validity treats. I nevertheless discuss some of the issues here to better frame the context and limitations of the study's findings.

The *selection* threat is concerned with factors that are not investigated in the study but that may affect the findings. In other words, does the study adequately rule out rival hypotheses [74]? In this study, the factors of concern would be influences other than (national) culture, such as organizational culture, team culture, time pressures, and domain expertise. These factors, like culture, may affect shared understanding in the distributed team. Although this study identifies two such intervening factors (i.e., communication mechanisms and the nature of the subcontract [to be discussed in Chapter 4]), I do not investigate the relative effects of these factors compared to culture.

The *selection of participants* threat addresses the bias in subject choice. In this study, due to resource limitations, I am only able to interview developers from CanTech. The findings therefore may be biased by a Canadian/CanTech perspective.

3.7.3 External Validity

External validity deals with the problem of proving that findings are generalizable beyond the immediate context. This study is an in-depth investigation of a relatively new field of research. The goal of the study is not to develop empirically generalizable findings. There are therefore many threats to the study's external validity and I have made few attempts to remove or alleviate them. The major threat is the study scope. The findings are based on one software development project, in one domain, with a specific outsourcing arrangement (i.e., subcontracting). This study does not attempt to validate the findings in other settings. Despite characteristics that are in common with many distributed projects (e.g., physical dispersion, limited communication, multiple organizations), I can not claim that the findings remain valid for all distributed projects.

3.7.4 Reliability

Reliability ensures that another investigator, following the same procedures and examining the same case, would arrive at the same findings. Given this definition, one prerequisite is to document in detail the study procedures [65]. In this thesis, I create a case study database, which documents all the interview questions, interview transcripts, and detailed descriptions of the case setting and subjects.

3.7.5 Data Accuracy

Data accuracy concerns all of the validity issues discussed in the previous sections. Drawing from Yin's work [65], the study design includes several mechanisms that are meant to help ensure this. These mechanisms are summarized here.

All the interviews are audiotaped and transcribed verbatim in order to ensure the accuracy of the raw data. Another method used to ensure the overall data accuracy is *triangulation*, the process whereby "researchers search for convergence among multiple and different sources of information to form themes or categories" [76]. Triangulation can be done across data sources (i.e., subjects), theories, collection methods (e.g., interviews, observations), and investigators [76]. This study triangulates across multiple data sources. Questions about scheduling, processes, and overall coordination and communication issues in the team are asked of all the subjects.

CHAPTER 4
FINDINGS

In the following sections, the study's findings are presented. This chapter first presents findings that characterize shared understanding in the distributed software development team. I discuss the extent to which shared understanding is established among the distributed sites, regarding each of the issues underlined by the study constructs (i.e., processes, requirements, resource management, and others). I also discuss any strategies the team uses to build shared understanding, as well as the related challenges and issues. I then present the cultural factors found to influence shared understanding. Finally, several sections follow that summarize findings pertaining to other emerging, intervening constructs that influence the team's shared understanding. For example, the subcontracting arrangements in the case appear to significantly limit visibility into the remote sites. Throughout these sections, wherever appropriate, I summarize the findings in the form of propositions. I present the propositions along with the supporting evidence (including sample quotes) and the context in which they are found to be valid in this study. The propositions are meant to highlight potential problem areas in distributed software development and, more importantly, to assist project personnel in planning and risk management. Figure 4-1 illustrates how intercultural differences, along with the nature of the relationship and communication mechanisms in the team, influence the various aspects of shared understanding.

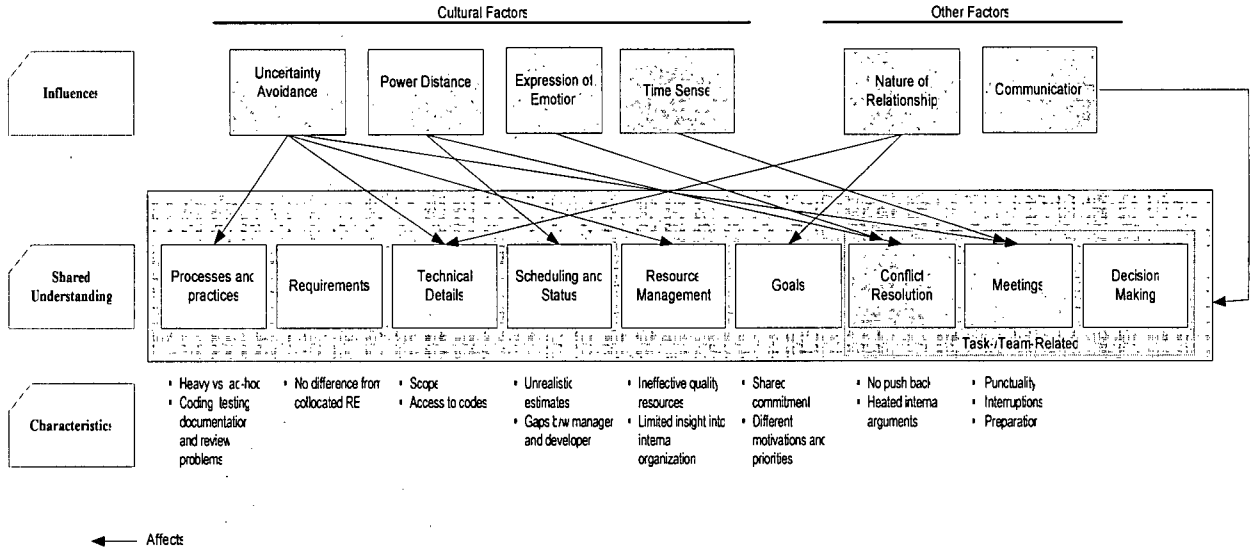


Figure 4-1 Effects of Culture on Shared Understanding

4.1 Shared Understanding in the Distributed Development Team

This section summarizes findings bearing on the characteristics of shared understanding in the distributed development team. The findings are organized by the problem areas of interest (as highlighted by the study constructs).

4.1.1 DEVELOPMENT PROCESSES AND PRACTICES

There is little, if any, shared understanding between the distributed sites regarding the development processes and practices. The problem is best illustrated by I-Sys and SofTar's divergence from the defined processes and CanTech's surprise at this discovery. Based on the process documents, CanTech expects that the project will use the processes as described. The CDR and various other reviews and audits, however, reveal that neither I-Sys nor SofTar actually follows the processes in their work. Through onsite interactions with the Italian developers, it becomes clear to CanTech that, at both development sites, there is little intention

to ever follow the processes. Some I-Sys and SofTar developers are not even aware of the existence of the process documents.

- *"There is a large disconnect between CanTech's expectation for process and the one that they put in place [...] We thought they were going to follow a much heavier process than they ended up doing ." (Jack)*
- *"I don't even think they knew what their own process documents were. I doubt that anyone have ever read them." (Colin)*

CanTech itself is a highly process-oriented organization and expects from its subcontractors a similarly rigorous adherence to the processes. The developers from CanTech are thus upset by I-Sys and SofTar's divergence from the written processes and consider this lack of processes the cause for the project's various quality and scheduling problems. I-Sys and SofTar, on the other hand, see little value in the processes and prefer a much more ad-hoc approach that is described by the CanTech developers as "putting out the biggest fire."

After the CDR, CanTech starts demanding that I-Sys and SofTar follow the processes. CanTech also starts sending its own developers to the remote sites more frequently. Some developers stay in Italy for extended periods of time and work closely with the Italian developers on tasks including requirements verifications and creating test plans. Others visit periodically for reviews and meetings. These efforts, nevertheless, seem to have little effect on changing the work style at the I-Sys and SofTar.

- *"Suggestions related to process and that they weren't really doing proper reviews, I doubt if they ever got fixed." (Jack)*
- *"I'm sure they still haven't modified their process. No doubt in my mind." (Blake)*

CanTech specifically differs from I-Sys and SofTar in its approaches to coding, documentation, testing, and various reviews. At CanTech, written standards are developed and produced to guide each of these activities. More importantly, the standards are enforced and followed within the

organization. At I-Sys and SofTar, although standards are also created, there is little adherence to the standards. As a result, the work of individual developers is constantly uncoordinated.

- *"They actually had the coding standards all beautifully documented and nothing wrong with the standards at all. But when we were doing the code walkthrough, it was absolutely clear that there was no adherence to the standard. So they had all the standards; they just didn't follow them as their normal process." (Colin)*
- *"One of [our] reviewers didn't like the names they were using for things - very unclear...But the notion of somebody questioning the names of things seemed very foreign to them. Even though coding standards talked about it and they had their coding standards, that somebody would care was a strange thing to them. I don't think they ever really understood why we were making - why some Canadian reviewers were making that comment." (Stephen)*

Stephen, having worked at both I-Sys and SofTar, observes that individual code sections are so different in their formats, documentations, and designs that a naïve reader could think the codes are produced for different projects.

- *"There were personalities to the different sections of the code. The ideal here for most projects is you try to have standards for the code so [...] although a team produces the code [...] you can't tell who produces which code except for the name in the comments. There each piece of code was very individualistic." (Stephen)*
- *"The formatting of the code and the code style, the style of the commenting, the format of the code, and the design of the code was pretty individualistic. If you were to take the extremes you could almost think they were different projects." (Stephen)*

With regard to documentation, the developers at I-Sys and SofTar are reluctant to produce written documents, especially those for planning or process-related purposes.

- *"In the contract we had a bunch of documents defined that they had to produce. They would try and get out of producing them or not produce what we wanted. They didn't mind producing analysis documents, when they had to analyze something. But any kind of documents related to planning or process or something like test plans or verification plans or verification reports, they just couldn't understand why we wanted that stuff, didn't want to do that, didn't think it had any value." (Blake)*

There is also a lack of review and coordination mechanisms for the documents. The same piece of information is often reproduced in a number of documents (with slight contradictions among them).

- *"What comes out of a document is a dump of information and that dump of information is not coordinated with the other guy's document. So he dumps out his information and it would be duplicated since now you've got two documents [that] covered sort of the same thing or something different. You could see lots of duplications in the documents. There was not a big coordinating entity that's reviewing what they were doing and sorting out the differences and making sure everything was consistently defined and neat [...] It's a lack of coordination and a lack of system-level review on the documents [...] They didn't see the value in kind of organizing all the information for us. They just wanted to kind of get it down on paper and let us sort it out. They didn't see it as their problem, whereas we would react to that and say 'well your documents aren't consistent. You've got to fix this; you've got to fix that.' And that's just not important to them." (Blake)*

Similarly, individual testers at I-Sys and SofTar are said to have different approaches of software testing. As well, regression testing, while a norm at CanTech, seems foreign and is deemed trivial by I-Sys and SofTar.

- *"Their concept of software testing was more 'put it through some ad hoc kind of thing so that the person who wrote the code is happy that it works, rather than following a structured and repeatable [process]. This is another issue that they had. The software group didn't think it was that important to be able to repeat the tests. They thought they would do the test once, with PA [Process Assurance] watching it, to the extent that the PA wanted to watch it, and that would be done. If there were bugs, they would fix the bugs, and the tests would show the bugs were fixed. But the idea of having to regression test over and over again wasn't something that they really bought into, even though we made them do it." (Stephen)*

Furthermore, the Italian testers consider CanTech's focus on the various technical reviews a waste of time and are unwilling to take part. Some managers at I-Sys also question CanTech's intentions for requesting the reviews.

- *"They didn't always see value in them [reviews] and felt that was us wasting their time, taking time away from them doing the job that we told them they weren't doing." (Holmer)*
- *"[The code walkthrough] certainly wasn't part of their normal process, although it was actually part of the [process] requirements. But I think they were very suspicious of the whole thing and tried to use code propriety and intellectual properties to stop us [from] really having that detailed [information].'" (Colin)*

4.1.2 System Requirements

Regarding the system requirements, there are no significant problems with shared understanding between the distributed sites. To define the requirements, CanTech presents I-Sys and SofTar

with the high-level requirements for the overall SBS system. The Italian sites, in turn, are responsible for deriving the software-specific requirements and producing formal, written Software Requirements Specifications (SRSs). CanTech regularly reviews the SRSs and provides feedback to I-Sys and SofTar for modifications. The developers consider such review-and-feedback mechanisms effective for resolving requirements issues and establishing shared understanding among the distributed sites.

- *"As far as requirements were concerned, there were specific requirements from CanTech for the SBS as a system, but there were no specific software requirements because it was considered they would derive requirements from the original SBS requirements." (Colin)*
- *Question: "Did I-Sys and SofTar have any inputs into the requirements as they were working on the project?" Answer: "I think the answer is yes because we gave them a higher level SBS requirements specification and they took that and they broke it down into subsystems specs of their own. So they did a lot of requirements analysis work and came up with specs and yes we reviewed them and found a lot of bugs and fixed those and had lots of reviews and lots of comments with that. If you're thinking that there's a disconnect because we said one thing and they said another thing, it wasn't really true because there was a lot of reviews up and down." (Jack)*

The CanTech developers do not find this (distributed) requirements engineering process any more problematic than their previous experience in collocated projects.

4.1.3 Technical Details

Although the distributed sites are reasonably successful at establishing shared understanding of the system requirements, challenges arise regarding understanding the system design and implementation. The CanTech developers admit that, prior to the Critical Design Review (CDR), they have little insight into the details of the technical work done at I-Sys and SofTar. As CanTech is not directly involved in the development, the organization has no direct access to the code and limited access to other technical artifacts. The little insight that CanTech has is mostly through the report-and-feedback mechanisms. I-Sys and SofTar regularly submit contractual and

process-defined reports to CanTech. After reviewing these documents, CanTech provides feedback for modification.

- *"We were not responsible for any of the hardware, software during this effort." (Jack)*
- *Question: "[Were you involved in] system design?" Answer: "It would be just an oversight role, reviewing and making sure. Our role as a subcontract manager on a technical side is really defining requirements, defining interfaces to the external things other than the SBS, and review the tests and make sure that the design meets the requirements, like there's objective evidence from the subs that they build in the things to our requirements. When we discover there are some problems, then we dig down deeper and that is essentially what we did. So that's really a digging of information out of the lower levels we did. Yeah, we are kind of like parasites on their development tasks, not really adding anything. We had value in the sense that from our side we understand what they're doing and hopefully can give them some useful comments. But at the end of the day they have to succeed themselves." (Jack)*

Specifically, CanTech has very limited knowledge regarding the system *scope*. After the CDR, through code walkthroughs and audits, CanTech learns that a number of requirements are left unimplemented. CanTech, relying only on the SRSs for the scope, is caught off guard that I-Sys and SofTar do not intent to implement the defined requirements. When questioned on the subject, the I-Sys and SofTar developers respond that they no longer consider the requirements necessary. The CanTech developers, while agreeing that re-scoping is necessary in software development, express surprise at not being involved in the decision making.

- *"When we did the review we said, 'Well, we'd like to see this function; where is it?' And they said, 'Oh we're not implementing that.' 'Why not?' 'Oh because we don't think it's necessary.' " (Colin)*
- *"[The re-scoping] wasn't huge. But it was quite a big surprise to us that we never really knew that they weren't really going to implement all this stuff, so we got quite upset about it." (Colin)*

It is interesting to compare I-Sys's and SofTar's approaches to deal with CanTech *feedback* regarding technical and requirements problems with those dealing with process-related feedback. While the I-Sys and SofTar developers readily accept specific technical-/requirements-related criticisms and take corrective actions, they are unwilling to act on CanTech's requests for more rigorous processes and practices. I thus conjecture that:

Proposition 1: It is easier to establish shared understanding for technical and requirements issues than for process issues.

4.1.4 Scheduling and Project Status

Scheduling is an area where the distributed sites have little shared understanding. CanTech often has difficulty tracking the project status at I-Sys and SofTar.

- *"We very often didn't catch onto what was happening." (Stephen)*
- *"It took a while to come to that realization [that I -Sys and SofTa]r are significantly late in the development." (Blake)*

As an example, the CDR, which elucidates the processes and quality problems at I-Sys and SofTar, does not take place until two and a half years into the project.

The problems partly have to do with the inaccurate estimates constantly presented by I-Sys and SofTar. The CanTech developers find the schedules, for the most part, implausible and, correspondingly, rarely met.

- *"They'd give us a schedule that you knew was just not going to happen based on past experience and based on what you knew the task ahead was. So they constantly did this. It was a constant attribute of the organization. They just could not give you a realistic schedule." (Jack)*
- *"The schedules weren't really achievable. It was pretty obvious but it was also obvious that their management didn't really see that." (Blake)*
- *"When I arrived they were due to deliver a piece of code within two weeks of my arrival. And six months later it was due to be delivered. It was still two weeks away and when I left it was just delivered." (Holmer)*

For instance, Stephen and Howard have to revise their duties after arriving at SofTar. The developers are originally sent to assist software testing. However, it becomes clear, after their arrival, that the development is not ready for testing. Stephen and Howard then respectively become involved in requirements and design verification. Their assignments are also extended due to delays in the project.

4.1.5 Resource Management

There is limited shared understanding between the distributed sites regarding the distribution and management of resources. First, CanTech has very little insight and influence into the internal organization at the remote sites. Initially CanTech is not aware that I-Sys hires SofTar as a secondary subcontractor for the project. Furthermore, CanTech has no direct control over the distribution of work and resource assignments at the remote sites. CanTech again relies on formal report-and-feedback mechanisms for managing the remote resources. There are often disagreements between CanTech and I-Sys/SofTar over how resources should be arranged for the project.

Particularly problematic is agreeing on the management and use of process-/quality-control personnel. As required by the process documents, I-Sys and SofTar institute process-/quality-control departments. The CanTech developers, nevertheless, find these resources have little active involvement in the project. The process-/quality-control personnel are described as *"lip service"* and *"completely ineffective."*

- *"They would have quality people there but [...] it was really lip service. I mean they weren't effective, completely ineffective." (Holmer)*
- *"They have people that are supposed to be project schedulers and managers. And they just really are just kind of outside of the main work and they don't have any real authority. It's kind of a visage, you know, so they meet the letter of what's required in the contract but it's not really part of their real process." (Blake)*

4.1.6 Goals

Shared commitment is observed from all the development sites involved. The CanTech developers recognize that the developers at I-Sys and SofTar strive to see the project succeed as much as those at CanTech. In spite of their differences in attitudes and approaches, the Italian developers are described as committed and hardworking.

- *Question: "So I guess they weren't really motivated to follow the process, but do you think they were motivated to see the system succeed as much as CanTech?" "Absolutely. Absolutely. They just wanted to do it their way." (Holmer)*
- *"They were honestly trying to satisfy us." (Stephen)*
- *"There was no malicious intent to miss the schedule. No they tried their hardest." (Jack)*
- *"They were definitely committed to getting the job done. They had a pride." (Holmer)*

The motivation factors, however, seem to differ. The development of SBS is based on an existing system that I-Sys and SofTar had previously developed. Also, the goal of I-Sys and SofTar, besides completing the SBS project, is to develop the system into a commercial-off-the-shelf product line. Other than the system-level requirements from CanTech, I-Sys and SofTar try to incorporate another set of their internal requirements into the system. CanTech, on the other hand, is soley concerned with the success of the SBS system. These differences result in some disagreements over system features and priorities.

Proposition 2: In the subcontracting relationship, the distributed sites are likely to differ in their goals and motivations.

4.1.7 Other Task-Related and Team-Related Issues

CanTech also differs from I-Sys and SofTar in their attitudes and approaches towards a number of other task- and/or team-related issues. These issues are not specific to any development activities but are regularly present in the project, affecting the interactions among the developers and the overall coordination of the distributed team.

4.1.7.1 Conflict Resolution

The developers discuss two types of conflicts in the interviews: conflicts among the distributed sites and conflicts within the local, immediate site. Several CanTech developers express high regard for how I-Sys and SofTar handle the pressure and demands from CanTech. The I-Sys and SofTar developers remain civil and friendly throughout the project, even when they are under constant and strict criticisms from CanTech for process and quality problems.

- *"As a group they were very good about taking the criticism and trying to act on it the way they understood it, and coming back maybe with an understanding that CanTech is very difficult and Canadians are very critical and hard to please. But they weren't closing doors to us because of it. They seem to just keep coming back with more." (Stephen)*
- *"They were very polite, very civil to us. You know we'd kind of be bashing over their heads routinely over the schedule and the process and anything else. They were always very professional in dealing with us." (Blake)*

The CanTech developers compare the way I-Sys and SofTar handle conflicts with how CanTech would react under similar conditions, asserting that CanTech would likely *"kick them [I-Sys and SofTar] out of the door"* had the roles be reversed.

A comparison is also made with another CanTech subcontractor based in Montreal:

- *"I think it was better that we had it [the memo incident] with [I-Sys] than some other people. For example when we did a lot of work with [a Montreal subcontractor], if we had done that to them, I think they would have never spoken to us again." (Colin)*

On the other hand, I-Sys and SofTar have somewhat different approaches for handling internal conflicts that do not directly involve CanTech. These approaches also exhibit drastic differences from how CanTech handles its internal conflicts. One developer notes that, at I-Sys and SofTar, it is not unusual for disagreements in meetings to become highly heated arguments, whereas at CanTech, developers remain physically calm even during the worse disagreements.

- *"That's another difference between Canada per se. Canada in the office people would very rarely yell at each other. It's culturally wrong here; it's not acceptable. There it's quite common. In a meeting people could start shouting at each other. The next day they would be fine. There's no obvious ongoing animosity. Whereas let's say here if people have strong disagreements, they may not voice it in Canada but the disagreement may go on and on silently for a long time." (Stephen)*

Such fiery arguments do not appear to affect the relationship between the Canadian and the Italian sites. The CanTech developers discuss being surprised and uncomfortable when they first witness the augments. However, they do not consider their working relationship with I-Sys and SofTar to be affected.

Proposition 3: Differences in conflict handling have little impact on the rapport and coordination of the distributed team.

4.1.7.2 Decision Making

A related issue is decision making. The managers at the Italian sites have much more authority than the CanTech managers. At I-Sys and SofTar, the developers rarely question the managers' decisions, even when they disagree. Compared to the CanTech managers, the managers at I-Sys and SofTar are more likely to overrule the developers' opinions and dictate decisions.

- *"The managers give commands and the people doing the work don't question them even if they disagree, to a certain extent." (Stephen)*

One developer compares this organizational hierarchy at I-Sys and SofTar with the relationship at CanTech:

- *"In CanTech you have different levels of engineers and stuff like in general [...] you know you have your organizational structure but at the working level it's pretty flat.[...] At I-Sys things were very, you know 'I'm here. I report to this guy. This guy reports to that guy. And this guy can command this guy to do something.' It was a lot more based on organization that way, rather than being flat. I wouldn't say I-Sys was kind of a flat working organizational structure at all." (Howard)*

Similarly, the I-Sys and SofTar developers rarely question inputs from CanTech. One developer notes that the unwillingness of I-Sys and SofTar to *push back* against the customer lead to a number of incorrect decisions that are only discovered and corrected later in the project.

- *"They never pushed back on us as much as we push back on our customer for sure [...] Now looking back I think we would've been better off from a technical point of view if [I-Sys] had pushed us back harder. We thought we knew what we wanted and we wanted them to do something and they half-heartedly fought against it and we pushed them again." (Colin)*

I conjecture that:

Proposition 4: When the developers are unwilling to push back against the management and the customer, it is more likely that technically incorrect decisions will be implemented in the project.

4.1.7.3 Meetings

CanTech differs greatly from I-Sys and SofTar in how meetings are conducted. Initially developers from CanTech are often frustrated by this difference. After travelling to Italy for a prescheduled meeting, the CanTech developers find it nonetheless difficult to get the accountable personnel to attend the meeting, at the predefined time, and for the entirety of the meeting. Meetings at I-Sys are often delayed for days. Interruptions are common as the attendees (and sometimes presenters) leave to attend to other duties.

- *"I can remember having people come over from CanTech. Maybe three fairly senior managers came over from CanTech for a meeting with I-Sys. You arrange the room and you arrange the meeting and get everybody ready. No I-Sys people. Why not? Because their bosses called them to a meeting. You know, who cares that three people, three senior CanTech managers have traveled over from Canada for a meeting? The boss called them in. They're not there. We had to wait. I think there was one occasion where we had guys over and it was the next day before they could meet." (Holmer)*
- *"The other thing is their lack of punctuality." (Colin)*

There are also differences in terms of the preparation for the meetings. At CanTech, each meeting has a predefined agenda. Attendees are notified of the preparation work required and are expected to have completed the assignments prior to the meetings. On the contrary, the developers from I-Sys and SofTar often arrive at the meetings unprepared, causing frustration for the CanTech developers.

- *"They would often show up to meetings unprepared. They didn't do their homework, so to speak. A lot of time even if it was a follow-up meeting, you'd expect you have your follow-up meeting and you'd have all your action items and stuff from the previous meeting done. You know they were done so you'd actually have some stuff to discuss. But it was actually quite entertaining that a lot of the time when you would have a meeting and you would sit there and you'd go 'OK. Now*

so we had this action item. What were the results?' And they're like, 'Oh. Oh let me go do that right now!' So that happens a few times." (Howard)

- *"They always amaze me how absolutely disorganized they were. I mean their manager would go into the meeting without any knowledge of the facts of what the change was being asked for." (Colin)*

4.2 Cultural Factors

This section summarizes the cultural factors found to influence the various aspects of shared understanding discussed in the previous sections. Differences in uncertainty avoidance lead to disparate approaches towards the development processes, scoping the system, resource management, and meeting protocols. Differences in power distance affect shared understanding regarding the project progress. Different expressions of emotions and the attitudes towards time also impact the developers' interactions.

4.2.1 Uncertainty Avoidance

Significant differences in uncertainty avoidance are observed between the Canadian and the Italian developers. The attitudes and approaches of the Canadian developers towards software development suggest a much lower tolerance for uncertain, unplanned situations than the Italian developers. The differences degrade the team's shared understanding regarding the development processes, technical activities, resource management approaches, and meeting protocols.

In general, the Italian developers see little value in tasks that aim to reduce uncertainty and ambiguity. While eager to work on analysis and implementation tasks, the Italian developers are reluctant to be involved in planning, documentation, and verification. They view these activities as a waste of effort and as slowing down the project rather than advancing it.

- *"I think culturally they are not inclined to do a lot of rigorous, regimented project management; it's against their nature." (Blake)*

- *"They don't view the work as methodical. I think it's too boring for them to be regimented with the schedule like that [...] I'm sure they're capable of estimating. But they always think they can achieve more, better than what's on the paper. So I think they're very motivated [...] but they don't like to impose a lot of project management on what they are doing, even though on paper they say that there'll be a project management control that kind of stuff. The reality is they don't do it, at least not in the same way we view it." (Blake)*

The Canadian developers, on the other hand, consider these tasks as essential for managing complexity and ensuring product quality. They rigorously follow the defined development processes, putting great emphasis on reviews and verification, and are often frustrated by the Italian developers' lack of process. I conjecture that:

Proposition 5(a): The development site with high uncertainty avoidance is likely to follow a heavy process, whereas the site with low uncertainty avoidance is likely to deviate from the process.

Proposition 5(b): The development site with low uncertainty avoidance may be reluctant to perform planning, documentation, and verification tasks.

Specifically, the Canadian and the Italian sites differ in their attitudes towards the meaning and use of *plans*. At CanTech, formal documents serve as a basis for constructing and guiding technical activities. Plans have great binding effects: they are followed in actual work. At the Italian sites, conversely, plans have little binding effects on future activities. In this case, the process documents are produced as part of the contract, but their usefulness, to the Italian developers, extends only to when the documents are reviewed and agreed to by CanTech. The Italian developers are surprised that CanTech in fact expects the plans (processes) to be followed.

- *"They wrote this wonderful process document and said this is what we do and CanTech says "Yes! This is just what we would like to see, just what we do! This is fabulous!' But once they've*

written it that's all they thought they had to do – 'Oh you mean you expect us to follow it?' And there's a huge gap there. That's where the real surprise came in. I actually had people say to me 'But you didn't expect us to follow that, did you?'" (Holmer)

Moreover, at CanTech, progress is evaluated precisely against the plan. When the plan is not met, the developers are expected to offer explanations as to why not. As well, any contingency strategies need to supply justifications (i.e., explaining how the new plan would succeed while the previous one did not). At I-Sys and SofTar, however, planning involves few feedback mechanisms. The approach is described as "if a plan doesn't work, make a new plan; if it doesn't work, make a new one."

Because of the different attitudes towards plans, the Canadian and the Italian sites are unable to established shared understanding regarding the development processes, system requirements, and the distribution of resources. For each of these issues, I-Sys and SofTar devise written plans that are reviewed by CanTech. For CanTech, these plans are expected to help build understanding and insight into the remote sites. It is CanTech's expectation that activities at I-Sys and SofTar will be carried out, more or less, according to the agreed-upon plans. However, because of the Italian sites' disregard for plans (a characteristic of low uncertainty avoidance), CanTech's assumption proves to be inaccurate. The Italian sites make little attempt to follow the processes and reject a number of the defined requirements. Their process-/quality-control personnel, though identified in the plans, are found to have little active involvement in the project.

I conjecture that:

Proposition 6: When the distributed sites differ in uncertainty avoidance, formal, written plans are ineffective as a tool for communication and building shared understanding.

In fact,

Proposition 7: When the distributed sites differ in their concepts of plans, relying on written documentation hinders timely detection of project problems.

Prior to the CDR, CanTech relies primarily on process-required plans and documents for monitoring the project progress at the remote sites. Viewing plans as highly binding, the Canadian developers are caught off guard when they discover the I-Sys and SofTar are not following the plans.

Furthermore, the differences in uncertainty avoidance result in different meeting protocols. While the Canadian developers devote much effort to prepare for meeting, the Italian developers conduct meetings in a more ad hoc manner with little preparation.

4.2.2 Power Distance

Hierarchy is another area in which great differences are observed between the Canadian and the Italian sites. The developers find the organization at I-Sys and SofTar much more hierarchical than that at Can Tech.

- *"I find like a large difference, distance between their management and the actual engineers who are actually doing the work." (Colin)*
- *"It's definitely the man at the top of I-Sys – whatever he said people do. So if your boss says do something you do it so it's very hierarchical in that sense." (Holmer)*

While the Italian developers exhibit greater deference to their management, the Canadian developers are more likely to voice their questions and disagreements.

- *"They don't really challenge their management. So like if their management says something that's like the law to them. Whereas with us we have much more freedom to challenge what our management is saying. I mean if my management says something I don't agree with I'd say 'I don't agree with it; I think we should have at least some discussion about it.' They seem to really respect like hierarchy, which I was quite surprised because I didn't realize that about Italians at all. But they're definitely very respecting of hierarchy. So you know [a technical lead] won't really go against his manager or the people who he is giving reports to, and the people below him as well. There's like a lot of chain of command, and a lot of respect between like the different layers of management." (Colin)*

I conjecture that:

Proposition 8: When working with a remote site with high power distance, it is difficult to gain insight into the project if communication is limited to the management.

This high power distance likely prohibits CanTech from gaining insight into the state of the development at I-Sys and SofTar, thereby affecting the team's ability to establish shared understanding of issues related to scheduling and project status. The CanTech developers note that there is significant disconnect between the *real* project status and what the management at I-Sys and SofTar present. Based on their experience working with the Italian developers, the CanTech developers find it unlikely that the Italian developers agree with their management with respect to the project status and estimates. Furthermore, through direct onsite interactions, the Canadian developers find their Italian counterparts straightforward in discussing the challenges of the project.

- *"I always get the feeling they're more than friendly and more than honest. So they are very honest in dealing with you [...] I didn't get any inclination that that was any sort of negative relationship. So they always told us -- I always thought they were more honest than they needed*

to be. Like they told us about some big problem and I thought well I wouldn't tell anybody that kind of stuff (laughing). No I think I never got any sort of negative vibes." (Jack)

As observed by the Canadian developers, it is more likely that at the Italian sites the schedules are dictated by the management. In other words, the developers, though having better knowledge, have little input into the plans, schedules, and reports presented to CanTech.

- *"The schedules [produced by the developers] were being overruled at different levels." (Jack)*
- *"There's definitely something [managerial pressure] to it [scheduling]. The management were demanding a certain schedule and the technical lead was complying with it [even though he knew it to be impossible]." (Colin)*

This respect for hierarchy at the Italian sites also affects their interactions with CanTech. As noted by the Canadian developers, many of the plans and schedules are produced to *please the customer*, rather than to reflect the project status. As well, the Italian sites rarely question the CanTech input and decisions. This unwillingness to push back against the customer leads to a number of technical errors that are discovered and corrected later in the project. I conjecture that:

Proposition 9: When the development site has high power distance, the developers are unlikely to question and push back against the customer and the management.

This proposition is also supported by Hofstede's work on power distance cultures [51] .

4.2.3 Expression of Emotions

The Canadian and the Italian developers differ in their expressions of emotions (and, consequently, conflict handling approaches). While emotions are expressed openly (sometimes through heated arguments) at the Italian sites, the Canadian developers have a much more neutral approach whereby feelings are kept controlled and subdued at the workplace.

Nonetheless, the Canadian developers also note the Italian sites' ability to quickly recover from conflicts. This forgive-and-forget attitude is best illustrated by I-Sys's reaction after the memo incident, in which a CanTech internal memo criticizing the Italian sites' progress is released to I-Sys. Upon reading memo, an I-Sys manager expresses strong objections to Holmer, the CanTech onsite resident. Other Italian developers are withdrawn in their interactions with the Canadian developers onsite. But after CanTech issues an apology to I-Sys, the Italian developers quickly recover from the incident and are again willing to work with the CanTech developers. Several CanTech developers express respect for the Italian sites, conjecturing that had the incident involved another subcontractor the relationship would likely be damaged permanently.

4.2.4 Sense of Time

The Canadian and the Italian developers also differ in their concepts of time. The differences result in disparate approaches to time management (that are best illustrated in meetings.) Specifically, the Italian developers are considered polychronic, viewing time as continuous and flexible. To the Italian developers, meetings can be flexibly delayed and interrupted. The Canadian developers, with a more monochronic perspective, perceive time as structured and purposeful. Each meeting has its predefined starting time, length, and agenda. Participants are notified prior to the meeting of such information. Participants are expected to arrive at the meeting at the predefined starting point and under normal circumstances stay at the meeting for its entire period.

4.3 Other Factors

Besides the cultural factors, analysis reveals a number of other contextual factors that have significant impact on the team's shared understanding. These factors are not included in the original construct lists and are not the focus of this study. They nevertheless emerge from the data as bearing significance and must be discussed.

4.3.1 Nature of Relationship

The subcontracting arrangements between CanTech and I-Sys/SofTar appear to have a significant impact on the team's shared understanding. CanTech hires I-Sys and SofTar as external subcontractors for the development of the SBS system. The contractual requirements limit the insight CanTech has into the technical activities and artifacts at the Italian sites. The Canadian and the Italian sites also appear to have different goals for the project. These differences consequently lead to different priorities and approaches towards the development.

As the prime in the subcontract, CanTech initially has no access to the code (and little to other technical artifacts) produced at I-Sys and SofTar. As CanTech can only monitor the project progress through reviews of formal process deliverables, problems in the project often go unnoticed.

- *"It took a while to come to that realization because as a prime we didn't have a lot of resources to look at every detail of what they were doing and we had people onsite and those people could sort of get inside into what's going on but still they threw up a bit of a wall. They don't want us to know exactly where they were at. They don't want us to interfere what they were doing so we really had to knock that wall over forcibly, like going through all this audits and walkthroughs." (Blake)*

After the code walkthroughs and audits reveal the various process, quality, and scheduling problems in the project, CanTech demands to put in more resources at both I-Sys and SofTar,

and to have more direct control over the technical activities. However, because these requests are in violation of the initial contract, I-Sys, at times, resists such CanTech presence. There is consequently frustration at both of the Canadian and the Italian sites.

As discussed in Section 4.1.6, CanTech also differs from I-Sys and SofTar in their ultimate goals for the project. The differences lead to some conflicting approaches and priorities.

4.3.2 Distance and Communication

The physical dispersion of the team and the lack of face-to-face communication significantly limit overall shared understanding among the distributed sites. A number of developers discuss the problems with synchronous communication channels such as teleconferences and videoconferences. While the developers find such technologies reasonably satisfactory for managerial discussions, discussing detailed technical issues remains challenging. The developers agree that it is much more effective to establish shared understanding, especially regarding issues related to the requirements, technical work, and resource distribution when the interaction is face-to-face. I conjecture that:

> **Proposition 10: Face-to-Face communication is more effective than mediated communication for establishing shared understanding, especially regarding technical issues.**

The benefits of face-to-face communication over computer-mediated channels are twofold. First, the CanTech developers find it easier to locate the right resources when they are onsite. During teleconferences and/or videoconferences, the CanTech developers interact mainly with managers

at I-Sys and/or SofTar. It is often difficult to find the particular developers accountable for the issues in discussion. While onsite, the CanTech developers have more direct access to and interactions with the Italian developers and therefore can gain a much more thorough understanding of the current project progress, challenges, and needs.

On the other hand, contextual cues that are integral for effective communication are often lost through mediated technologies. As a result of losing cues such as facial expressions and eye contact, the CanTech developers find it difficult to detect misunderstandings and disagreements.

- *"It's very easy to think that actually their communication is better than what it really is. So it's very easy to think that they understand, when actually you find out later that they were not understanding completely. It's very hard – I found anyway – to pick it up during a videoconference or a teleconference, because you basically get a nod or yes or something, but without actually seeing the person look confused and trying to pick up on that is quote difficult. So it's very easy for them to say 'yes, we understand,' but when you actually get there you realize they did not understand." (Blake)*

Often, such misunderstandings and disagreements only become known when they manifest in actual defects or delays in the submitted artifacts, resulting in rework and delays.

4.4 Summary of Findings

To summarizing the study findings presented in this chapter, I revisit the original research questions first outlined in section 1.2.

The first research question concerns the extent to which shared understanding is established in the distributed development team. The findings of this study suggest that, in general:

Proposition 11: The distributed development team has little shared understanding.

Not only the developers often lack visibility into the technical activities, artifacts, and progress at the remote sites, but they often differ significantly in their development processes, goals, and priorities. At the beginning of the project, shared understanding is particularly lacking, even when the distributed sites all have strong domain expertise. To add to the problem, the developers are unaware of this lack of shared understanding and caught off guard when learning that the remote sites do not meet their expectations. I conjecture that:

Proposition 12: The distributed team assumes a greater extent of shared understanding than there really is.

The second research question addresses how shared understanding is developed among the distributed team. In the case study, CanTech initially relies mostly on formal review-and-feedback mechanisms for gaining control and insight into the project. Echoing Bass's [24] claims, such artifact-driven approaches prove to be ineffective. Planning-related documents do little to establish shared understanding among the distributed sites, while documents of a reporting nature are incapable of timely uncovering problems in the project. Later in the project, the team develops a stronger shared understanding, as the developers start to have more direct, face-to-face interactions. Activities such as reviews, audits, and stationing proxy developers at the remote sites also help uncover project problems and align the distributed sites in their approaches and knowledge.

The third research question deals with the effects of intercultural differences on the development of shared understanding. The findings support that:

Proposition 13: Intercultural differences limit shared understanding in the distributed team.

In particular, the study finds that:

- Differences in uncertainty avoidance lead to misalignment among the distributed sites regarding the development processes and practices
- High power distance may cause that only the managerial perspective (rather than the perspective of the developers working on the floor) is communicated in meetings.
- Differences in how emotions are expressed in the workplace lead to different conflict handling approaches.
- Differences in the perception of time manifest in different meeting protocols.

Of the abovementioned cultural factors, issues related to uncertainty avoidance, power distance, and time sense cause a great deal of frustration and confusion in studied development team. On the other hand, different conflict resolutions, contrary to intuition, do not lead to significant problems. Although the data collected in this study does not offer a straightforward explanation, the difference may have to do with the relative *visibility* of these cultural issues. Expressions of emotions (e.g., shouting and fist-clenching in the meetings) are readily visible. The developers therefore can make quick observations and adjustments. Differences in uncertainty avoidance, hierarchy, and sense of time, on the contrary, are less visible and only come to light when they manifest in actual problems (e.g., divergence from the processes, delays, poor quality). Fixing these problems (after the fact) then requires more effort and cost, therefore causing much frustration and/or even conflicts.

Of the predefined cultural constructs, the study finds little evidence on the effects of collectivism/individualism and communication patterns on shared understanding. Overall, the study confirms to current literature in observing that intercultural differences affect (often negatively) the coordination in the distributed software development team. The findings go further than the literature in explaining specifically how various cultural factors influence aspects of shared understanding and development activities.

4.5 Limitations of the Study

A number of issues prevent the study from more thoroughly investigating the research questions. The first is the study scope. The findings are based on one development project, in one domain, with a specific outsourcing arrangement (i.e., subcontracting). There have been no efforts to validate the study's findings in other settings. I make no claims about the empirical generalizability of the findings.

Another limitation is, due to limited resources, only the developers from CanTech are interviewed. As discussed in Section 3.6, the findings therefore may have a CanTech and/or Canadian bias. Collecting and analyzing data from the I-Sys and SoftTar/Italian developers may shed new light on the findings.

Furthermore, I do not use any specific metrics to measure the extent of shared understanding, the effects of cultural diversity, and the overall coordination of the development team. As such, although the findings have established that intercultural differences limit shared understanding in the distributed environment, I am unable to empirically evaluate:

- the specific effects of shared understanding on the overall team coordination,

- the effects of different cultural dimensions on shared understanding, compared to other detriments such as distance and communication, and
- the extent of shared understanding in the distributed team, compared to a collocated team.

However, it must be noted this study is exploratory. The goal is not to produce any deterministic profile of culture for the distributed development team, but to highlight potential problem areas and to offer culturally-sensitive perspectives on the problems. The study findings do not make causal inferences or predictions.

CHAPTER 5

IMPLICATIONS FOR THE PRACTITIONER

Table 5-1 List of Propositions

General:

- **The distributed development team has little shared understanding. (P11)**
- **The distributed team assumes a greater extent of shared understanding than there really is. (P12)**
- **Intercultural differences limit shared understanding in the distributed team, especially differences in uncertainty avoidance, power distance, expression of emotions, and sense of time. (P13)**

Uncertainty Avoidance:

- **The development site with high uncertainty avoidance is likely to follow a heavy process, whereas the site with low uncertainty avoidance is likely to deviate from the process. (P5(a))**
- **The development site with low uncertainty avoidance may be reluctant to perform planning, documentation, and verification tasks. (P5(b))**
- **When the distributed sites differ in uncertainty avoidance, formal, written plans are ineffective as a tool for communication and building shared understanding. (P6)**
- **When the distributed sites differ in their concepts of plans, relying on written documentation hinders timely detection of project problems. (P7)**

Power Distance:

- **When the developers are unwilling to push back against the management and the customer, it is more likely that technically incorrect decisions will be implemented in the project. (P4)**
- **When working with a remote site with high power distance, it is difficult to gain insight into the project, if communication only with the management. (P8)**
- **When the development site has high power distance, the developers are unlikely to question and push back against the client and the management. (P9)**

Expression of Emotions:

- **Differences in conflict handling have little impact on the rapport and coordination of the distributed team. (P3)**

Nature of Relationship:

- **In the subcontracting relationship, the distributed sites are likely to differ in their goals and motivations. (P2)**

Communication:

- **Face-to-Face communication is more effective than mediated communication for establishing shared understanding, especially regarding technical issues. (P10)**
- **It is easier to establish shared understanding regarding technical and requirements issues than that regarding process issues. (P1)**

The contribution of this study is a set of propositions, which are presented in Chapter 4 and summarized in Table 5.1. The propositions characterize shared understanding and the effects of culture in distributed software development. The propositions are based on findings observed in this study and are meant to highlight problem areas, regarding such issues as development processes, requirements engineering, technical details, resource management, project scheduling, and other related concerns (e.g., conflict resolution, decision making, and meeting protocols). The propositions further focus on the effects of cultural diversity on shared understanding in the team. From the propositions, combined with studies in the literature, I draw a number of implications. The implications include strategies and techniques that developers and project personnel may consider in planning distributed development projects. No efforts have been made to empirically validate the strategies, techniques, and practices suggested here.

P11 and P13 together suggest that, overall, the distributed development team has very little shared understanding, especially at the beginning of the project, and that this lack of shared understanding, in part, has to do with the intercultural differences among the developers. Even more problematic, as suggested by P12, is that the distributed sites are often oblivious to the disconnects in their approaches and attitudes. In this case study, CanTech mistakenly believes that shared understanding is established (especially regarding the development processes), based on the process documents and contractual requirements. Nevertheless, because of culturally different concepts of plans, in reality the team has little shared understanding (as suggested by P6). The distributed sites are caught off guard when they realize their *mismatches in expectations*. As a lesson learned, one developer points out that it is important to, especially at

the beginning of the project, **assume there is no shared understanding**, no matter how many documents and contracts are reviewed and signed.

To alleviate the problems associated with differing definitions of plans, one technique is to **interview** the specific developers about their experiences and knowledge. For instance, instead of asking the subcontractor to submit a written verification plan, one may ask the accountable developers to explain their personal experiences in software verification.

Formal documents are also ineffective (and sometimes even misleading) as a monitoring mechanism (as suggested by P7). P4, P8, and P9 further suggest that it is difficult to gain insight into the activities and problems at the remote site if interacting with the management only. In distributed work, it is of great importance to visit the remote site and have **direct, face-to-face interactions with the developers** doing the work (as suggested by P10). Agile practices such as scrums may help establishing shared understanding by providing a direct communication channel among the developers doing the work. Holding frequent, regular **reviews** also helps establish shared understanding in the team.

P5 highlights the problems associated with differences in uncertainty avoidance. Developers with low uncertainty avoidance may be reluctant to follow a heavy process and, in particular, resist planning, documentation, and verification tasks. As suggested by P1, it is extremely difficult and costly for an external force to change the internal working of an organization. While CanTech feels some of their efforts, such as **stationing onsite proxy developers**, help bringing some rigor to the process at I-Sys and SofTar, most of their efforts are futile. In retrospective, the

CanTech developers consider it important to prioritize and identify critical areas for improvement – *pick your fights*, in other words.

Different meeting protocols lead to much frustration for the distributed team in case. As a mitigating strategy, CE, the CanTech resident at I-Sys starts scheduling meetings to be at the start of the week. She reasons that, as such, there is still the possibility of having the meeting within the week, even if delays occur. Other CanTech developers adjust to the Italian time management approach by bring their work into the meeting, while waiting for other attendees to show up.

Some of the techniques suggested above may be difficult to implement in a subcontracting relationship (as suggested by P2). As such, it is important to **anticipate worst-case scenarios and corresponding contingency plans** in the contract. For example, it may be better to send a number of proxies to the remote sites at the start of the project (and remove them later, depending on the project progress), than to only start sending proxies when the project goes off the rail.

In general, when entering the distributed project, it is important to be aware of the cultural differences and anticipate mismatches in attitudes, approaches, priorities and knowledge. As noted by one of the CanTech developers, "*you just have to change your expectations*" and devise mitigation and contingency plans.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

The subject of this thesis was a qualitative case study on a distributed software development project. Described here were the findings of the study, the motivation and context for this work, the related literature, and the detailed research design of the study. The concepts of interest investigated by the study were shared understanding in the distributed team and the effects of intercultural differences. I found that the all of the studied aspects of shared understanding (development processes and practices, technical requirements, technical details, resource management, scheduling and status, and goals), as well as related issues such as conflict handling, decision making, and meeting approaches, were affected by intercultural differences. Specifically, the influencing cultural dimensions included uncertainty avoidance, power distance, time sense, and expression of emotions. The study found no evidence that differences in collectivism/individualism and communication contexts affected shared understanding. The study further identified two non-cultural influences, communication mechanisms and the nature of the relationship, as affecting shared understanding in the distributed development team.

The study was an exploratory effort and offered one angel of understanding about transient, implicit coordination in global software development. The goal was to develop a set of propositions that (a) provide detailed insight into an area where little previous research has been done, (b) advise project planning and risk management activities, and (c) inform research in the area. Many other, more complex questions are yet answered and require further investigation. There are many areas in which future research efforts could be conducted. First and as a

continuum to this study, efforts should be made to interview developers from other development sites (i.e., I-Sys and SofTar) in order to reduce bias. As discussed in Chapter 3, the study described in this thesis is part of a larger research agenda involving multiple cases. Future research efforts should validate the propositions found in this case with other cases or through other collection methods (e.g., surveys). Developing empirical metrics that more precisely evaluate the impacts of culture and the effects of shared understanding (in relation to other non-cultural influences), both in the distributed team and in the collocated team would strengthen the generalizability and applicability of the findings. Future research will also benefit from incorporating other research (including quantitative) methods, such as surveys, archival analysis, and ethnography. These methods would help uncover more grounded and general (but not deterministic) findings and theories about the phenomena of shared understanding and culture in global software development. Finally, future research may more specifically examine the effects of intercultural differences on various development practices, particularly the more *social* ones that require close interactions among the developers (e.g., pair programming, inspection meetings).

REFERENCES

[1] T. W. Malone and K. Crowston, "The Interdisciplinary Study of Coordination," *Computing Surveys*, vol. 26(1), pp. 87-119, 1994.

[2] J. D. Thompson, *Organization in Action*. Chicago: McGraw Hill, 1960.

[3] A. van De Ven, A. L. Delbecq, and R. Koenig, "Determinants of Coordination Modes within Organizations," *American Sociological Review*, vol. 41(2), pp. 322-337, 1976.

[4] R. L. Deft, "Organizational Information Requirements, Media Richness and Structural Design," *Management Science*, vol. 32(5), pp. 554-571, 1986.

[5] S. M. Schmidt and T. A. Kochan, "Conflits: Towards Conceptual Clarity," *Administrative Science Quarterly*, vol. 17(3), pp. 359-370, 1972.

[6] B. Victor and R. S. Blackburn, "Interdependence: An Alternative Conceptualization," *Academy of Management Review*, vol. 12(3), pp. 486-498, 1987.

[7] K. Crowston and E. E. Kammerer, "Coordination and Collective Mind in Software Requirements Development," *IBM Systems Journal*, vol. 37(2), pp. 227 - 245, 1998.

[8] J. A. Cannon-Bowers and E. Salas, "Reflections on Shared Cognition," *Journal of Organizational Behavior*, vol. 22 pp. 195 - 202, 2001.

[9] D. R. Ilgen, J. R. Hollenbeck, M. Johnson, and D. Jundt, "Teams in Organizations: From Input-Process-Output Models to IMOI Models," *Annual Review of Psychology*, vol. 56 pp. 517-543, 2005.

[10] K. Crowston, "A Coordination Theory Approach to Organizational Process Design," *Organization Science*, vol. 8(2), pp. 157-175, 1997.

[11] B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," *Communications of ACM*, vol. 31(11), pp. 1268-1287, 1988.

[12] J. D. Herbsleb and R. E. Grinter, "Architectures, Coordination, and Distance: Conway's Law and Beyond," *IEEE Software*, vol. 16(2), pp. 63-70, 1999.

[13] J. D. Herbsleb and D. Moitra, "Global Software Development," *IEEE Software*, vol. 18(2), pp. 16-20, 2001.

[14] R. E. Kraut and L. A. Streeter, "Coordination in Software Development," *Communications of ACM*, vol. 38(3), pp. 69-81, 1995.

[15] D. B. Walz, J. J. Elam, and B. Curtis, "Inside a Software Decision Team: Knowledge Acquisition, Sharing, and Integration," *Communications of ACM*, vol. 36(10), pp. 63-77, 1993.

[16] Y. Hsieh, "Culture and Shared Understanding in Distributed Requirements Engineering," presented at 2006 IEEE International Conference on Global Software Engineering, Florianopolis, Brazil, 2006

[17] "The Chaos Report," The Standish Group, West Yarmouth, MA 1995, 2001.

[18] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining Mental Models: A Study of Developer Work Habits," presented at 2006 International Conference on Software Engineering, Shanghai, China, 2006

[19] Y. Hsieh, E. MacGregor, and P. Kruchten, "Matching Expectations: When Culture Wreaks Havoc with Global Software Development," *Submitted to Journal of Architectural and Planning Research*, 2007.

[20] P. N. Robillard, "Opportunistic Problem Solving in Software Engineering," *IEEE Software*, vol. 22(6), pp. 60-67, 2005.
[21] K. V. Beaman, "Myths, Mistiques, and Mistakes in Overseas Assignments: The Role of Global Mindset in International Work," *IHRIM Journal*(November/December), pp. 40-53, 2004.
[22] S. L. Jarvenpaa and D. E. Leidner, "Communication and Trust in Global Virtual Teams," *Organization Science*, vol. 10(6), pp. 791-815, 1999.
[23] P. N. Robillard, "The Role of Knowledge in Software Development," *Communications of the ACM*, vol. 42(1), pp. 87-92, 1999.
[24] M. Bass, "Monitoring GSD Projects via Shared Mental Models: A Suggested Approach," presented at Workshop on Global Software Development for the Practitioner, collocated with 2006 International Conference on Software Engineering, Shanghai, China, 2006
[25] K. E. Weick and K. H. Roberts, "Collective Mind in Organizations: Heedful Interrelating on Flight Decks," *Administrative Science Quarterly*, vol. 38(3), pp. 367 - 381, 1993.
[26] W. J. Orlikowski, "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing," *Organization Science*, vol. 13(3), pp. 249-273, 2002.
[27] E. Carmel and R. Agarwal, "Tactical Approaches for Alleviating Distance in Global Software Development," *IEEE Software*, vol. 18(2), pp. 22-29, 2001.
[28] S. Whittaker, "Theories and Methods in Mediated Communication," in *Handbook of Discourse Processes*. Florham Park, NJ, 2003.
[29] M. Cusumano, A. MacCormack, C. F. Kemerer, and W. Crandall, "A Global Survey of Software Development Practices," *eBusiness @ MIT*, 2003.
[30] E. Carmel, *Global Software Teams*. Upper Saddle River, NJ: Prentice Hall, 1999.
[31] G. Dafoulas and L. Macaulayk, "Investigating Cultural Differences in Virtual Software Teams," *EJISDC*, vol. 7(4), pp. 1-14, 2001.
[32] D. E. Damian and D. Zowghi, "Requirements Engineering Challenges in Multi-site Software Development Organizations," *Requirements Engineering Journal*, vol. 8 pp. 149 - 160, 2003.
[33] J. Hanisch, "Understanding the Cultural and Social Impacts on Requirements Engineering Processes - Identifying Some Problems Challenging Virutal Team Interaction with Clients," presented at The 9th European Conference on Information Systems, Bled, Slovenia, 2001
[34] D. W. Karolak, *Global Software Develoment: Managing Virutal Teams and Environments*. Piscataway: IEEE Computing Society, 1998.
[35] S. Krishna, S. Sahay, and G. Walsham, "Managing Cross-cultural Issues in Global Software Outsourcing," *Communications of the ACM*, vol. 47(4), pp. 62-66, 2004.
[36] E. MacGregor, Y. Hsieh, and P. Kruchten, "Cultural Patterns in Software Process Mishaps: Incidents in Global Projects," presented at Workshop on Human and Social Factors of Software Engineering, collocated with 2005 International Conference on Software Engineering, St. Louis, Missouri, 2005, ACM Press.
[37] G. M. Olson and J. S. Olson, "Mitigating the Effects of Distance on Collaborative Intellectual Work," *Economics of Innovation and New Technology*, vol. 12(1), pp. 27 - 42, 2003.
[38] G. Walsham, "Cross-cultural Issues in Global Software Outsourcing," *Retrived from http://www.almaden.ibm.com/institute/pdf/2004/Geoff_Walsham_1.pdf*, 2004.

[39] G. Borchers, "The Software Engineering Impacts of Cultural Factors on Multi-Cultural Software Development Teams," presented at 2003 International Conference on Software Engineering, Portland, Oregon, 2003
[40] G. Walsham, "Cross-Cultural Software Production and Use: A Structurational Analysis," *MIS Quarterly*, vol. 226(4), pp. 359-380, 2002.
[41] J. Hanisch and B. Corbitt, "Requirements Engineering during Global Software Development: Some Impediments to the Requirements Engineering Process - a Case Study," presented at 12th European Conference on Information Systems, Finland, 2004
[42] D. E. Damian and D. Moitra, "Global Software Development: How Far Have We Come?" *IEEE Software*, vol. 23(5), pp. 17-19, 2006.
[43] R. Klimoski and S. Mohammed, "Team Mental Model: Construct or Metaphor," *Journal of Management*, vol. 20(2), pp. 403-437, 1994.
[44] J. A. Cannon-Bowers, E. Salas, and S. A. Converse, "Shared Mental Models in Expert Team Decision Making," in *Current Issues in Individual and Group Decision Making*. Hillsdale, NJ: Erlbaum, 1993, pp. 221-246.
[45] J. E. Mathieu, T. S. Heffner, G. F. Goodwin, E. Salas, and J. A. Cannon-Bowers, "The Influence of Shared Mental Models on Team Process and Performance," *Journal of Applied Psychlogy*, vol. 85(2), pp. 273-283, 2000.
[46] G. C. Wilson, *Supercarrier*. New York: Macmillan, 1986.
[47] J. A. Espinosa, R. K. Kraut, S. A. Slaughter, J. E. Lerch, J. D. Herbsleb, and A. Mockus, "Shared Mental Models, Familiarity, and Coordination: A Multi-method Study of Distributed Software Teams," presented at 23rd International Conference on Information Systems, Louisiana, New Orleans, USA, 2002
[48] L. L. Levesque, J. M. Wilson, and D. R. Wholey, "Cognitive Divergence and Shared Mental Models in Software Development Project Teams," *Journal of Organizational Behavior*, vol. 22 pp. 135 - 144, 2001.
[49] C. B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering*, vol. 25(4), pp. 1-14, 1999.
[50] E. T. Hall, *Beyond Culture*. New York, NY: Anchor Books, 1976.
[51] G. Hofstede, *Culture and Organizations: Software of the Mind*: McGraw-Hill, 1997.
[52] R. Linto, *The Cultural Backfround of Personality*. New York, NY: Appleton Century Co., 1945.
[53] A. Montagu, *Statement on Race: An Annotative Elaboration and Exposition on the Four Statements on Race Issed by the United Nations Educational, Scientific and Cultural Organization*, 3rd ed. New York, NY: Oxford University Press, 1972.
[54] C. Roberts, E. Davies, and T. Jupp, *Language and Discrimination*. London, UK: Longman, 1992.
[55] H. Spencer-Oatey, *Culturally Speaking: Managing Rapport through Talk across Cultures*. New York: Cassel, 2000.
[56] S. Dahl, "Intercultural Research: The Current State of Knowledge," Middlesex University Business School Discussion Paper, 2004.
[57] G. Hofstede, *Culture's Consequences: International Differences in Work-Related Values*. Beverly Hills, CA: Sage, 1980.
[58] F. Trompenaars and C. Hampden-Turner, *Riding the Waves of Culture: Understanding Diversity in Global Business*: Nicolas Brealey Publishing, 1996.

[59] L. Laroche, *Managing Cultural Diversity in Technical Professions*. Burlington, MA: Butterworth-Heinemann, 2002.
[60] B. MacSweeney, "Hofstede's Model of National Cultural Differences and Their Consequences: a Triumph of Faith - a Failure of Analysis," *Human Relations* pp. 89-118, 2002.
[61] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Eman, and J. Rosenberg, "Prelinimary Guidelines for Empirical Research in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 28(8), pp. 721-734, 2002.
[62] V. R. Basili and S. Elbaum, "Better Empirical Science for Software Engineering," presented at 2006 International Conference on Software Engineering, Shanghai, China, 2006
[63] C. B. Seaman, "Organizational Issues in Software Development: An Empirical Study of Communication," in *Department of Compter Science*: University of Maryland, 1996.
[64] Y. Dittrich, M. John, J. Singer, and B. Tessem, "CFP: Information and Software Technology special issue on Understanding the Social Side of Software Engineering: Qualitative Software Engineering Research," 2006.
[65] R. K. Yin, *Case Study Research: Design and Methods*, vol. 5, 2 ed. Thousand Oaks, California: Sage Publications, 1994.
[66] J. D. Brewer, "The Research Process in Ethnography," in *Ethnography, Understanding Social Research*. Philadephia: Open University Press, 2000, pp. 57 - 193.
[67] M. B. Miles and A. M. Huberman, *Qualitative Data Anaysis: An Expanded Source Book*: Sage Publications, 1994.
[68] K. M. Eisenhardt, "Building Theories from Case Study Research," *Academy of Management Review*, vol. 14(4), pp. 532-550, 1989.
[69] J. Corbin and A. Strauss, "Grounded Theory Research: Procedures, Canons, and Evaluative Criteria," *Qualitative Sociology*, vol. 13(1), pp. 3-21, 1990.
[70] B. G. Glaser, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago, IL: Aldine Publishing Co., 1967.
[71] A. Coffey and P. Atkinson, *Making Sense of Qualitative Data: Complementary Research Strategies*. Thousand Oaks, CA: Sage Publications, 1996.
[72] J. L. Gibbs, "Loose Coupling in Global Teams: Tracing the Contours of Culture Complexity," in *Faculty of Graduate School (Communication)*. Los Angeles: University of Southern California, 2002.
[73] K. M. Eisenhardt, "Politics of Strategic Decision Making in High-Velocity Environments: Toward a Midrange Theory," *The Academy of Management Journal*, vol. 31(4), pp. 737-770, Bourgeois, L. J.
[74] S. Easterbrook and J. Aranda, "Case Studies for Software Engineers," presented at ICSE'06, Shanghai, China, 2006
[75] S. M. Cox, "Transcription of Recorded Interviews." University of British Columbia, 2006.
[76] J. Creswell and D. L. Miller, "Determining Validity in Qualitative Inquiry," *Theory into Practice*, vol. 39(3), 2006.

APPENDIX A – DATA COLLECTION

A.1 Interview Protocol

Logistical Information: record the subject's name and date of the interview

Script for the beginning of the interview:

- Inform the subject of the content of the consent form
- If appropriate, have the subject sign the consent form
- Ask for permission to tape record the interview
- If permitted, turn on the tape recorder

Background Information:

- Describe the project:
  - What is the system being developed?
  - How is the project distributed?
  - In total, how many people are involved in the project?
  - From each site, how many people are involved?
  - What functionality does each site provide?
  - What is your role in the project?
  - How long is your participation?

Process and Practices:

- What development process is used in the project?
- Do the development sites all use the same process?
- Do you have written process documentation?
- Do you practice any of the Agile processes (e.g., XP, Scrum)?
- Over the course of the project, do you ever have to change or adapt the process and any practices? Do the remote site(s) have to change or adapt their process?

Requirements:

- What is the requirements engineering process?
- Do the distributed sites negotiate on the requirements?
- Are there misunderstanding of requirements? Discuss.

Communication:

- What communication medium do you use (e.g., email, instant messaging, telephone, teleconferencing, face-to-face meetings)?
- How effective are the communication medium?
- What is the purpose of communication (e.g., distributing information, making decisions)?
- Who do you communicate with? Are there designated proxies?
- How often do you communicate with the remote site(s)?
  - Ad hoc communication?
  - Regular communication?

- Who drives the interaction?
- Are there any communication mishaps?
- How formal/informal is the atmosphere?

Scheduling:
- How is the schedule determined? Who decides the schedule?
- Are the schedules met?
- Do you think everyone think the same way about the importance of the schedule?
- What happens when the schedule is not met?

Documentation:
- What is the purpose of documentation?
- Does documentation reflect the reality of the project?
- Do you have documentation standards? Are they followed (by all the sites)?

Enabling Technologies:
- Do you use any specialized ICTs for collaboration?
- Does every site use the same ICTs?
- Is there any resistance from any site against any particular ICT?

Collaboration:
- How do you decide on:
  - Division of work
  - System design
  - Use of technologies
  - Resource management and distribution?
- How much collaboration is involved in design, implementation, testing, and other development activities?
- How much insight do you have into the activities at the remote sites?

Resources:
- How do you decide on the team structure:
- Are there problems with getting the resource you need from the remote sites?

Trust:
- How would you describe the relationship among the distributed sites?
- How would you characterize the level of trust among the distributed sites?
  - Do you trust the other site(s)?
  - Do you feel you are trusted?
  - How does that compare with the level of trust in your immediate team?
- Does the level of trust change over the course of the project?
- Are there any explicit attempts to build up trust?
- How would you draw the boundary of your team?

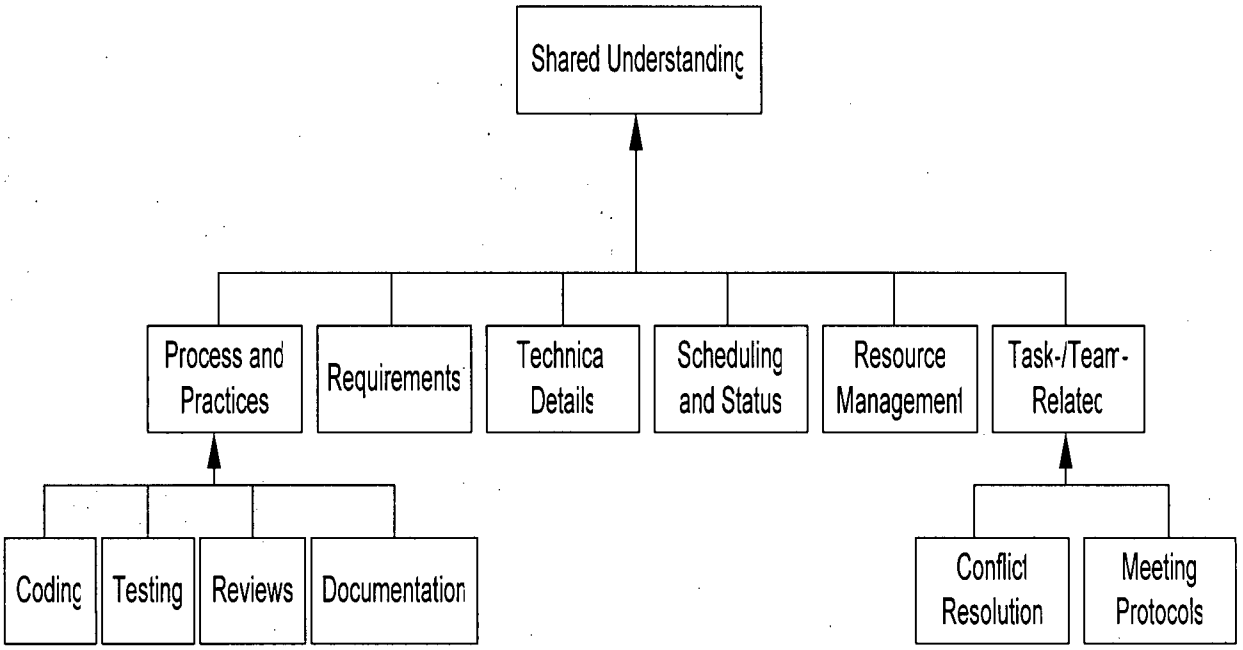
Miscellaneous Issues:
- Do you notice any differences between your site and the remote site(s) regarding:

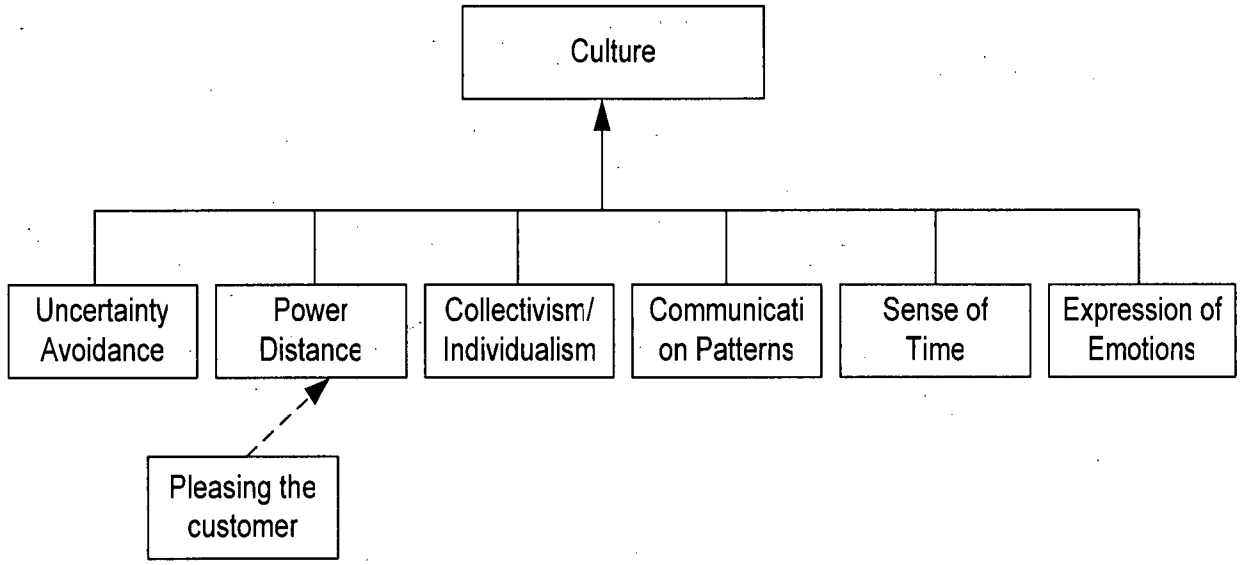
    - Conflict resolution
    - Change management
    - Attitudes towards planning
    - Hierarchy in the organization?
- Can you recall any significant mishaps or conflicts?
- How do you rate the commitment and motivation level at each of the development sites? Are there significant differences?
- Do you use any "bridgeheads" in the project? What are the roles of these bridgeheads? How are the bridgeheads perceived and treated at the remote site(s)?
- What do you consider to be the biggest challenge in the project?
- Any other interesting experiences or thoughts about the project?

A.2 Code List with Frequencies

Code	Number of Occurrences
Shared Understanding	16
Process and Practices	123
Requirements	8
Resource Management	13
Technical Details	46
Schedules and Status	81
Task- and Team-related Issues	6
Shared Goal	16
Culture	11
Collectivism/Individualism	3
Communication Patterns	2
Expression of Feelings	28
Power Distance	38
Time Sense	16
Uncertainty Avoidance	40
Coding	12
Documentation	8
Testing	5
Reviews	16
Communication	45
Face-to-Face Communication	13
Conflict Handling	44
Meeting Protocols	10
Time Management	10
Nature of Relationship	22
Onsite Proxies	10
Rapport	24
Pleasing the customer	8
Trust	7
Lessons Learned	38
Domain Expertise	1
Plan	49

A.3 Code Networks





is a dimension of is related to

A.4 Sample Codes and Quotations

Code(s)	Sample Quotes
SH – Process and Practices Uncertainty Avoidance	"I'm not sure [the process] was ever built into their team. It wasn't just that they were not doing it because of schedule pressure and things like that. I just don't think that was part of their culture." (Colin) "They are just not inclined to do a lot of rigorous, regimented project management. It's against their nature." (Blake) "CanTech in general tends to be a lot more by-the-book really, following step-by-step configuration and rules, while on the other hand the I-Sys guys and SofTar guys tend to be a little more freewheeling." (Howard)
SH – Process and Practices Plans	"They let us a little bit strayed. They wrote a process document that said a lot of good things but it didn't work out that way. So they said they were gonna do a lot more testing and a lot more design documentation, and it really, really didn't happen that way." (Jack) "I don't think they knew what their own process documents were. I doubt that anyone had ever read them." (Colin) "They saw [the process] as an ideal and this is what we need to tell our customers. It's not something that anyone would expect anyone really to follow. It's like theoretical and it's on the shelf there." (Colin)
SH – Process and Practices Coding	"It was obvious they had these documents of software coding standards. You asked them. It took them a month to dig them out of some outside file chamber and clearly no one ever read those documents." (Colin) "If you were to look at the different pieces of work that different people were doing, for example if you were reviewing the code...there were personalities to the different sections of the code. The idea here for most project is you try to have standards for the code so although a team produces the code, you can't tell who produces which code except for the name in the comments. There each piece of code was very individualistic." (Stephen)
SH – Process and Practices Documentation	"Documentation was not one of their strong points. They didn't like to produce documentation. We'd always fight with them to produce documentation. In the contract we had a bunch of documents defined that they had to produce. They would try and get out of producing them or not produce what we wanted. So it was a real struggle. They didn't mind producing analysis documents, when they had to analyze something and present the results. But any kind of document related to planning or process or something like test plans or verification plans or verification reports, they just couldn't understand why we wanted that stuff, didn't want to do that, didn't think it had any value." (Blake)
SH – Process and Practices Reviews	"I don't think they were too happy about [our request for reviews]. I guess the review wasn't part of their normal process." (Colin) "They didn't have much respect for these reviews. To them it was just a waste of time; they had to get all this information together; they had to sit in meetings, more meetings with CanTech to be beaten up again." (Holmer)

Code(s)	Sample Quotes
SH – Process and Practices	"[All the testers] had different approaches and styles of software testing." (Stephen) "Their concept of software testing was more 'put it through some ad hoc kind of thing so that the person who wrote the code is happy that it works,' rather than following structured and repeatable test plan." (Stephen)
SH – Technical Details Nature of the Relationship	"They just generated documents for us to look at. We didn't have access to the code." (Jack) "They had previously done a lot of software development but we just had no visibility in it whatsoever because it was done on a different program." (Colin)
SH – Technical Details Uncertainty Avoidance Plans	"When we started to do the walkthroughs we discovered that a lot of the implementation didn't agree with their own requirements. So there was a lot of functionality in their requirements that they actually never intended to implement. Now this was obviously quote foreign for us. And one of the reasons why we never really found out was we were just seeing the requirements and we were happy that the requirements had the functionality that we required and they were all nicely traced back to our requirements." (Colin)
SH – Resource Management Uncertainty Avoidance	"They really do pay lip service to it. They have people that are supposed to be project schedulers and managers. They really are just outside of the main work and they don't have any real authority. It's a visage so they meet the letters of what's required in the contract. But it's not really part of their process" (Blake) "They would have quality people there but it's lip service. I mean they weren't effective, completely ineffective." (Holmer)
SH – Scheduling and Status Power Distance	Interviewer: "Do you think [pressure from the management] might have been part of what the engineer was coping with when they were trying to do the schedules?" Respondent: "Definitely. There's definitely something to it. I mean his management were demanding a certain schedule and he was complying with it." (Colin)
SH – Scheduling and Status Power Distance Pleasing the Customer	"It's a direct object of this wanting to please mentality." So they'd give us a schedule that you knew was just not going to happen, based on past experience and based on what you knew the task ahead was. So they constantly did this...They wanted to please CanTech and CanTech kept telling them to bring it in sooner." (Jack)
SH – Goals Nature of Relationship	"They were in a large way reusing existing software. So they had a different approach. They wanted to try and reuse what they already had, in some way resisting what we wanted to do whereas we had the approach of [we wanted] something done and we didn't care how much they had to change their software to do that." (Colin)
SH – Goals	Interviewer: "Do you think they were motivated to see the project succeed as much as CanTech?" Respondent: "Absolutely. Absolutely. They just wanted to do it their way." (Holmer)
Meeting Protocols Sense of Time	"A guy was presenting in a formal review. He used to take cell phone calls in the middle of his presentation – "Excuse me" – and walk to side for 10 minutes chatting on the phone and then come back and carry on with the presentation." (Colin)

Code(s)	Sample Quotes
Meeting Protocols Uncertainty Avoidance	"If we're going to meetings, we've got 2000 slides and four dry-runs and a huge amount of effort for preparation, especially for big meetings. But even more smaller meetings we generally have the idea 'if we're going to meetings, we have to be prepared.' At I-Sys, it's like a meeting is just another excuse to drink some coffee and have a chat. There's no fixed agenda." (Colin)
Conflict Resolution Power Distance	"They never pushed back on us as much as we push back on our customer for sure...I think if they feel the customer is higher from a hierarchical point of view, they'll tend to agree with the customer instead of pushing back against them." (Colin) "Their program manager would crack the whip quite a bit more than what you might see here." (Howard) "The managers give commands and the people doing the work don't question them even if they disagree, to a certain extent." (Stephen)
Conflict Resolution Expression of Emotions	"The Italian guys got upset and they waved their hands about it but it didn't last very long. I forgave us quite easily afterwards whereas I think a lot of other companies would have really held a grudge for longer." (Colin) "On the spur of the moment, on the actual crux of the moment, [the team lead] lost control completely. I've never seen that happen here." (Colin)
Communication	"The real important things we managed to resolve were always done during onsite meetings." (Colin) "At first we took for granted that they understood [in videoconferences]. We would repeat ourselves and we assumed that went through. And we realized that wasn't happening all the time. So we became – we assumed that they didn't understand. So we would follow up with them again with email." (Blake)

A.5 Sample Memos

Memo on Jack-Transcript 1

A) **In the subcontract case, there is no shared understanding between the two sites regarding the development process and practices.**

a. All 12 segments coded with *Shared Understanding of Process and Practices*, either explicitly or implicitly, indicate that (1) I-Sys and SofTar did not follow the development process in the manner that CanTech wanted them do, and (2) this divergence from the process came as a surprise to CanTech.

b. Segment 1, 2, and 3 coded with *Shared Understanding of Process and Practices + Plan* explains that the gaps in shared understanding of the process and practices might have to do with the process document submitted to CanTech by I-Sys and SofTar. Specifically, I-Sys and SofTar presented to CanTech a process document as part of their contract. CanTech assumed that the process that was specified in the document would be used in the project. But to I-Sys and SofTar, (1) the purpose of the document was more to please their client, CanTech, than to assist themselves in the project (Segment 2), or at least, (2) there seemed little binding between what was written in the document and what needed to be executed.

c. The two segments coded with *Plan + Culture* suggest that these disparate attitudes towards the use, significance, and value of plans may be culturally based.

d. Overall, based on these segments and codes, it seems that,

Proposition 1: In the subcontract relationship, the usefulness/effectiveness/value of a plan is degraded when the distributed sites have (possibly culturally) different ideas as to what the plan means in the project. Consequently there are gaps in shared understanding (on the development process and practices, in this case).

Documentation alone is insufficient for ensuring shared understanding of the process and practices. More rigorous investigation is needed.

B) **Shared understanding of the technical requirements seems present in the subcontract case.**

C) Shared understanding of the technical work seems present in the subcontract case.

a. Segments coded with *Shared Understanding of Requirements and Shared Understanding of Technical Work* seem to suggest that, although CanTech did not have direct access to the details of the work done at I-Sys and SofTar due to their contract, through tight review and feedback mechanisms, CanTech was able to gain understanding and insights into the technical work. (Note: But this contradicts the situation before CDR.) Jack does not seem to think that there were big problems with shared understanding of the technical aspects of the project.

Proposition 2: In the subcontract case, it is easier to develop shared understanding on technical issues of the project with reviews and feedback, than to develop shared understanding on the process-related issues. ***(Note: I don't know if I can really draw this conclusion from the data analyzed thus far.)***

Memo on Colin – Transcript 1:

A) Echoing Jack-Transcript 1, **there is no shared understanding between the two sites regarding the development process and practices.**

   a. Segment 2 – 10 and 15 coded with *Shared Understanding of Process and Practices* support the findings in Jack – Transcript 1, that I-Sys and SofTar surprised CanTech by not following the process as specified in the process document.

   b. In particular (Segment 2 and 6), the developers at I-Sys and SofTar did not follow the coding standards and were surprised when CanTech requested code reviews (although code reviews were included as part of the process per document).

   c. Segment 1 offers a further explanation as to why there were gaps in shared understanding of the development process and practices (other than the fact that CanTech made the assumption based on the process document):
      - The process document submitted by I-Sys and SofTar was based on the ESA standards, which CanTech knew and approved.

   d. The finding that "culturally different attitudes towards the use, significance, and binding effects of plans cause the gaps in shared understanding of the development process and practices (from Jack-Transcript 1) are also supported here.

B) Shared understanding of technical requirements:

   a. Colin explains how requires are derived and created in the project. It isn't exactly clear if there were problems with developing shared understanding of the technical requirements between the two sites, though there were no indications that there were.

   b. If combined with findings from Jack-Transcript 1, the segment on Shared Understanding of Requirements may be used to help further explain how shared understanding of requirements was developed and ensured in the project:
      - CanTech presented I-Sys and SofTar with the high-level requirements for the overall bus system, and I-Sys and SofTar were responsible for coming up with the software requirements.
      - Continuous reviews and feedback were used to ensure shared understanding of requirements (from Jack).

C) **There is *limited* shared understanding of technical work and project status in the subcontract case.**

   a. Colin's comments (Segment 2, 5 – 11) on shared understanding of the technical work somehow contradict Jack's findings. Specifically, I-Sys and SofTar re-scoped the system without involving CanTech and the changes came as a surprise to CanTech.

b. These gaps in shared understanding again might have to do with the two sites' (culturally) different attitudes towards plans. CanTech assumed that I-Sys and SofTar would implement the system according to the requirements documents. But I-Sys and SofTar "never intended to implement" some of the features, and (again) the documents were "developed for the customer but not really meant to be applicable."

c. Another issue at play is the lack of shared understanding regarding task-related knowledge (in Cannon-Bower's terms). CanTech expected to be involved in re-scoping activities and decisions, but I-Sys and SofTar did not seem to think so. These gaps in shared understanding of task-related knowledge may also be culturally based (Segment 1 coded with *Shared Understanding of Task-Related Knowledge + Culture*). In short,

Proposition 3: Gaps in shared understanding of task-related knowledge cause gaps in shared understanding in UPDATED knowledge of the technical detail and status of the project.

From this we may draw that over-reliance on documentation runs the danger of losing sight of the current status and progress of the project, especially if the distributed sites have different attitudes towards the significance and use the documents. In other words, when working with a different culture, it is of even greater importance to try and get the most updated information about the project, through means other than formal documentation.

D) **The limited/lack of shared understanding regarding the technical details, status, and resource management of the project are influenced by other factors (i.e., nature of the relationship, communication).**

a. Segments coded with Shared Understanding of Resource Management indicate that the gaps in this aspect were a consequence of the subcontracting relationship between CanTech and I-Sys/SofTar.

b. Segment 2 – 4 coded with Shared Understanding of Project Status indicate that the gaps in this aspect had to do, at least in part, with the distance between the two sites.
   - Shared understanding of the status is improved after CanTech started stationing people onsite and face-to-face interaction was more available.

E) **Shared understanding of the other players improves after developers from CanTech started to stay in Italy for extended periods of time.**

a. Rapport is developed (Segment 1-3)

b. Shared understanding of the other players improves as the developers start to gain understanding of the other players' cultural orientations (Segment 4). (?)