Requirements Change Management in Global Software Development: A Multiple Case Study

Waqar Hussain

A thesis submitted to Auckland University of Technology in fulfilment of the requirements for the degree of Doctor of Philosophy (PhD)

2016

School of Engineering, Computer and Mathematical Sciences

Auckland University of Technology

***Primary Supervisor*: Associate Professor Tony Clear**

Secondary Supervisor: Jim Buchan

***Third Supervisor*: Professor Daniela Damian**

***Mentor*: Professor Stephen G. MacDonell**

Abstract

This thesis reports a comprehensive investigation of the challenges encountered in managing requirements change and investigates the associated role of collaborative technologies in Global Software Development (GSD)[1].

Software has always been considered as malleable. Given this, changes to software requirements are inevitable during the development process. Despite many software engineering advances over several decades, Requirements Change Management (RCM) is a source of project risk that is especially important in GSD, particularly in today's context of rapidly evolving businesses and technologies. Although the effective management of requirements is a critical aspect of GSD, understanding of the challenges and practical approaches to their management are still a contemporary research issue. A key enabler of collaboration in GSD are the collaborative technologies available to geographically dispersed teams. It can be expected that these same technologies also address some of the challenges of RCM in a GSD context, but their role and efficacy is unclear. The repeated experience of dissatisfaction in managing requirements in global collaborations, in spite of the rich body of knowledge and available Collaborative Technologies (CTs) in place, have been the primary motivators for this work.

Two GSD cases (one in New Zealand and the other in Pakistan) are explored in depth through a case study methodology that applies a fit-for-purpose research framework, to analyze the challenges of RCM in GSD and the associated role of CTs. A corpus of data based on participant interviews, change-related process and project artefacts as well as observations from the two selected cases is analyzed in depth in this study, through the application of a thematic content analysis technique. Thus a very rich and firmly grounded understanding of RCM processes in practice and the role of CTs is developed.

This exploratory study has resulted in several initial conjectures that form the basis for novel theorizations. Conceptually, the results from this thesis are synthesized to represent the first known RCM process model that captures the practices of informal requirements change management - an under-theorized concept in literature. Substantively, the comparison and contrast of the two case studies have resulted in the identification of eight new challenges faced by those responsible for managing requirements change in globally

[1] GSD is defined as "development of a software artifact across more than one location" (Smite et al., 2014)

distributed development environments. Through the research findings, the bridging roles of client liaisons and proxies in requirements change management are described in a manner distinct from present understandings in GSD literature. Furthermore, while the research findings confirm the adequacy of CTs in use and their supportive role for those carrying out RCM activities in GSD, several obstacles imposed by CTs are also identified.

Table of Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Waqar Hussain

In loving memory of

my father, Mohammad Nisar Mirza.

Gone but not forgotten!

Acknowledgements

This thesis has benefited from the help and support from several people which deserve my respect and acknowledgement. I am grateful to everyone who has been directly or indirectly involved in this process.

First of all, I would like to thank to my supervisor team Associate Professor Tony Clear for his invaluable contribution to this thesis through insightful discussions, guidance and support throughout the process. I am also thankful to my secondary supervisor Jim Buchan for his guidance and advice as well as making the necessary resources available to support this research. I am also thankful to my third supervisor Professor Daniela Damian for her expert advice and feedback on my thesis.

I want to extend my deepest gratitude and appreciation to Prof Steve MacDonell who has been an active, foresighted and insightful mentor, expertly guiding and supporting me throughout this journey. Without his contribution this work would not have been completed.

In addition to my supervisors who were closely associated with this work, I would also like to express my gratitude to Prof Didar Zowghi for her timely guidance and advice on my research. She instilled a desire for and commitment to improve the quality of this thesis.

I must acknowledge the contributions of the large supporting cast from the organizations involved in this research; the people who were interviewed, allowed access to their work artefacts, speculated about the challenges they were facing as well as shared their views regarding the role of collaborative technologies. Without their support and co-operation there would have been nothing to report.

A special thanks to Abdul Rahim for his sustained support and co-operation throughout this research. He generously agreed to allow access to his organizational data and intelligently managed the work schedule of his staff and created opportunities for me to unobtrusively collect the required data.

It goes without saying that I am indebted to the faculty of Design Creative Technologies and School of Engineering, Computer and Mathematical Sciences (SECMS) for my PhD scholarship to pursue my studies at AUT. I also appreciate the employment opportunities provided by SECMS AUT, in the form of Lecturing, Project Supervision, Tutoring as

well as Research Assistant. Thanks to Anne Philpott, Dr Stephen Thorpe, Dr Gisela Klette for the excellent BCIS R&D project coordination which made it a great learning experience.

This thesis would not have been possible without the support of my family, especially my wife who has been consistently supportive of my studies and understood the demands imposed by PhD. Your continuous encouragement, support, prayers and love made the journey easy. Thank you.

Ethics Approval

This research work has been approved by the Auckland University of Technology Ethics Committee on 11 January 2013 AUTEC Reference Number 12/287 “Requirements change management in global software development”.

Chapter 1. Introduction

This thesis investigates the challenges faced by those who must manage requirements change during software development carried out across geographically distributed sites. This study has profiled the first known, in depth, multi-case case study exploring the lifecycle2 activities of a requirements change and the associated helpful and hindering role of collaborative technologies for two projects in global software development (GSD)settings. These projects fall under the offshore outsourcing category of GSD [3].

This chapter introduces the research work reported in this thesis. In Section 1.1 the background and importance of the area of research is presented. The specific motivations underpinning the research are provided in Section 1.2. Next, details of research aims are provided in Section 1.3 followed by the research objectives in Section 1.4. The research design is described in Section 1.5, where the research methods and analysis techniques are briefly introduced. A summary of the key contributions of the work is provided in Section 1.6 (detailed in Chapter 7), which follows the published works resulting from the research in Section 1.7. An outline of the thesis structure is provided in Section 1.8.

1.1 Research Background

Contemporary software systems are densely interwoven to the very fabric of our society and economy. Globalization of the software industry has pushed software development from local to global markets where practitioners increasingly work in distributed arrangements (Ebert, 2011). In fact, outsourcing with global software development ranks among the top business ideas of the past 100 years (Ebert, 2011, p. ix). This transition is largely driven by promising benefits such as economy of expenditure, reduced time to market and utilization of highly skilled global resources. The promised benefits that lead companies to explore global sourcing, however, come with a price of delay, loss of visibility, coordination and control (Sangwan, Bass, Lullick, Paulish, & Kazmeier, 2007). Despite its drawbacks, the trend of globalization in software development continues to gain momentum making requirements engineering (RE) increasingly distributed (Schmid, 2014). The popularity of software globalization is evident from the fact that

[2] This thesis considers the lifecycle activities of a change to include: *Elicitation, Analysis, Negotiation, Validation, Prioritization, Specification, Managing Uncertainty, Traceability, Testing and Implementation* based on ISO/IEC TR 24766:2009 framework (de Gea, Nicolás, Fernández Alemán, Toval, Ebert, & Vizcaíno, 2012).

[3] GSE is used as a synonym of GSD for the purposes of this thesis and offshore outsourcing is a strategy for "leveraging external third-party resources situated in a different country" (Smite et al., 2014)

global information technology spending on outsourcing has risen from $US 100 billion in 2010 (Ebert, 2011) to $US 442 billion in 2014 (Kanaracus, 2014).

The dynamic and competitive nature of modern business frequently triggers the evolution of business needs and gives rise to new and changing software requirements. Requirements Change (RC) is an accepted phenomenon in contemporary software development (Davis, 2013). In fact, change is welcomed and embraced in Agile development approaches as a means of adding value and improving usability. However, uncontrolled changes pose a risk to both the cost and the quality (McGee & Greer, 2012).

Managing changes in requirements is difficult enough for teams working together at one physical location even under the best of circumstances but it presents additional difficulties when stakeholders are globally distributed (Schmid, 2014). Distance adversely affects communication and aggravates coordination and control problems (Herbsleb, 2007). In such contexts communication among stakeholders about requirements engineering becomes less frequent and more constrained (Herbsleb, 2007).

The need for adequate tool support to reduce the impact of distance becomes critical in managing changes in requirements in GSD (Kelanti, Hyysalo J, Välimäki A, Kuvaja P, & M, 2013; Sabaliauskaite, Loconsole, Engström, Unterkalmsteiner, Regnell, Runeson, Gorschek, & Feldt, 2010). Requirements Management (RM) tools help in maintaining traceability, status and flow of requirements to align them with user needs as well as downstream activities (Sabaliauskaite et al., 2010). Experiences have shown that an appropriate end to end tool chain increases efficiency and success of distributed projects (Keil, Kuhrmann, & Niinimaki, 2011). As a result, tool development and integration has been on the prominent agenda of researchers and practitioners (Kelanti et al., 2013; Sabaliauskaite et al., 2010). A large pool of commercial and open source tools (over a hundred) is available to offer RM support for collaboration over requirements (Birk & Heller, 2015; de Gea, Nicolás, Fernández, Toval, Ebert, & Vizcaíno, 2015).

1.2 Motivation

Despite the popularity of GSD, the vast majority of global activities do not deliver to targets and waste billions of dollars (Ebert, 2011). Almost 20% of all outsourced projects are cancelled in their first year (Ebert, 2011) and 50-80% outsourcing marketplace projects fail downstream for not reaching their desired objectives (Jørgensen, 2014). More recent studies suggest that even 50% of all outsourced projects fail after the first year (Ebert, 2014).

Requirements Change Management (RCM) a key development support process has been identified as the primary cause of many such GSD failures (Bieg P, 2014, p. 4; Davis, 2013; Levin & Ward, 2011). Despite efforts through proposed technological and process-oriented solutions along with an extensive body of knowledge, research has yet to overcome the challenges faced by multi-site organizations in requirements and change management activities (Schneider, Torkar, & Gorschek, 2013). Poor management of the ever changing user requirements, therefore, continues to hurt organizations through missed deadlines, budget overruns and wasted resources (Bieg P, 2014; Lai & Ali, 2013; Mishra & Alok, 2011).

However, on a positive note, some researchers still believe that the ultimate success factor in GSD is knowing about requirements and managing them effectively (Ebert, 2011, p. 39). This research is therefore motivated by the expectation to reduce waste in outsourced projects by better understanding the existing RCM challenges and providing empirically-grounded insights and recommendations for the practitioners to address those challenges.

Although researchers have been investigating the area of RM in GSD for over a decade, there is relative paucity of research into managing lifecycle activities of a requirements change (Carlshamre & Regnell, 2000; Grehag, 2001). Similarly, tool support for managing requirements and change in GSD has also received substantial attention (Kelanti et al., 2013; Rodríguez, Vizcaíno, Ebert, & Piattini, 2010; Tell & Babar, 2012). However, their role in RE lifecycle activities has not been adequately explored. Some researchers have identified limitations of the existing tools and believe that activity based computing theory, as an alternate, could provide a solution to the existing challenges in GSD (Tell & Babar, 2012). Therefore, this research attempts to understand the actual role played (helpful or hindering) by collaborative technologies to ascertain if more sophisticated technological solutions are required to solve RCM challenges in GSD.

Despite the works just described and the plethora of tools available (Birk & Heller, 2015), there remains a shortage of rigorous and meaningful empirical research on how organizations perceive and support the critical process of requirements change management in GSD projects. Rather than immediately targeting tool development, this research takes a different approach - it adapts a lifecycle perspective to understand the challenges of managing requirements change in GSD environments. The research also explores the intertwined relationship of process, practice and the assumed role of collaborative technology in managing changes. This research posits that such an approach is required to address the issues faced in requirements change management because

currently they are not well understood, there is a lack of supporting empirical evidence for the proposed solutions and there are no theories available to resolve them.

Another motivation comes from the research itself, which suggests that the key to achieving success in GSD lies in overcoming challenges to RE activities (including change management) posed by GSD (Lopez, Nicolas, & Tov, 2009). Furthermore, managing risks (such as those posed by constantly changing requirements) throughout the lifecycle of GSD projects is expected to help in achieving the promised GSD benefits (Verner, Brereton, Kitchenham, Turner, & Niazi, 2014). Such a lifecycle approach is necessary to study requirements and change management practices which overlap with many other disciplines such as product and project management, configuration management and systems engineering (Davis, 2013, p. 35).

As GSD is a practical undertaking, the ultimate success of requirements engineering research depends on how relevant the results are to industry's short and long term needs (Cheng & Atlee, 2007). The lack of conversation between academic researchers and practitioners inhibits the understanding of issues and problems that industry struggles with and as a result may focus researchers' efforts on obscure or unimportant areas of research (Cleland-Huang & Damian, 2011). Substantial anecdotal evidence exists to suggest academia prescribed practices are not followed by practitioners in small organizations (Beecham, OLeary, Richardson, Baker, & Noll, 2013; Buchan, Ekadharmawan, & MacDonell, 2009). This study addresses this challenge and purposefully engages with practitioners to understand the dynamics of requirements change management (RCM) process, practice and collaborative technology's role in real life GSD settings.

Finally, if the challenges in GSD can be reduced, it can potentially benefit information technology industries in-general by minimizing wastage of resources within GSD projects. While such outcomes are understandably well beyond the bounds of this specific research effort they do lend further motivation to work of this nature.

1.3 Research Aim

The research addresses the limitations of prior work noted above and contributes an increased understanding of the challenges faced by practitioners and the perceived role of collaborative technologies in RCM. The aim of this research is to better support requirements change management in GSD by providing insights on i) the challenges faced

during the lifecycle activities of a change and ii) the helpful or hindering role of collaborative technologies during those activities.

1.4 Objectives

In order to achieve the research aim outlined above, the specific objectives of this study are as follows:

1. *To identify the current challenges faced by practitioners in managing requirements change in GSD environments.*
2. *To describe the impact of GSD challenges on the lifecycle activities of a requirements change process.*
3. *To establish the supportive as well as the hindering role played by collaboration technologies (CTs) in managing requirements change related activities in GSD from the current practices of CT use and the participants' perspectives.*
4. *To identify the gap between the literature and practices for managing requirements change in GSD.*
5. *To extend the current theories and body of knowledge in the area of managing requirements change in GSD*

1.5 Research Design

There are several ways to address the research objectives, however due to the nature of the interpretive study suggested in the research aim and objectives (outlined in Section 1.3) a qualitative research design was deemed appropriate to carry out this study (Creswell, 2013).

A case study approach is adopted in this research to gain an in-depth and first-hand understanding of the phenomenon under investigation that involves collaboration of people across multiple organizations who belong to different social and cultural backgrounds (Yin, 2006). The case study is carried out in a bottom-up style using an interpretivist approach (Klein & Myers, 1999).

The specific choice of a qualitative interpretive exploratory case study research methodology for this study is influenced by these factors: goodness to fit with the research objectives, the researcher's philosophical beliefs, time constraints, the nature of the phenomenon being studied and access to suitable data (Yin, 2003). A multisite qualitative case study, a variant of case study design (Yin, 2014, p. 18), was employed in this research

with the intention to provide robust evidence base and to strengthen the ability of case study design to generalize while preserving in-depth description (Herriott & Firestone, 1983).

Data analysis for this research was driven by the framework used by Briand, Basili, Kim and Squier (1994), for analysing requirements change related qualitative data. The framework is more recently adapted and used by Nurmuliani, Zowghi and Powell (2004) to study causes of requirements volatility in a GSD context.

Thematic analysis, a qualitative data analysis technique, is applied inductively because of the qualitative nature of data collected (through semi-structured interviews, artefact analysis and observations). A bottom-up approach is utilized to identify, analyse and interpret themes or patterned meaning in the qualitative data to provide insights for theory building (Braun, Clarke, Terry, Rohleder, & Lyons, 2014, p. 95). Identified themes are compared, contrasted and combined to form meta-level themes using qualitative observations. These meta-level themes are analysed based on their inter relationships to gain deeper insights about the phenomenon under investigation. Data triangulation from multiple sources such as interviews, artefacts and observations is utilized to provide multiple insights and strengthen the work under consideration as well as for drawing stronger conclusions (Creswell, 2013). This form of thematic analysis is applied in a recent study about cultural influence on testing practices in GSD by Shah, Harrold and Sinha (2014).

1.6 Key Contributions

The key contributions from this research are categorized based on the three domains (conceptual, methodological and substantive) proposed by McGrath (1983). This thesis traverses two of the three domains mentioned by McGrath with the intention of contributing to our knowledge about requirements change management in GSD. Accordingly, the following summary of contributions resulting from this study is grouped by domain.

1.6.1 Contributions in the Substantive Domain

1) This research adds to the very few multi case studies in the area of requirements change management in global software development.
    a. This research confirms thirty-seven known challenges of global software development and identifies eight new challenges related to requirements

change management carried out in two globally distributed development environments.

2) The research describes the impact of the six newly-identified problems of global software development on the activities related to requirements engineering and change management for each of the studied cases.

3) The research addresses a gap in the literature by exploring collaborative technology support for lifecycle activities of requirements change management in global software development.

4) The research elaborates on the central role of spreadsheets as CTs that facilitate carrying out requirements change management activities.

1.6.2 Contributions in the Conceptual Domain

5) This study profiles the first known, in depth, multi-case case study exploring the lifecycle activities of a requirements change for two GSD projects carried out in professional environments.

6) The research produces the first known model that captures in one place the sites, roles, activities, artefacts and collaborative technologies of GSD projects by adapting an Activity Based Model and notation taxonomy for Global Requirements Engineering (GRE).

7) The research produces an extended model to represent GSD challenges and their impact on RCM and RE activities by adapting the model proposed by (Damian & Zowghi, 2003b).

8) The research presents a model of a requirements change management lifecycle synthesized from two RCM models in practice from the two studied cases in GSD contexts. It has the potential to be applicable in contexts similar to those studied in this research.

9) This research proposes a novel concept of "informal requirements change" verified by both cases studied and validated by the practitioners involved in the case study.

10) This research presents a unifying and synthesizing conceptual model of RCM based on the two GSD project contexts studied in this research.

11) The research presents several implications (Chapter 7), for the benefit of practitioners involved in global software development.

1.7 Publications Related to This Thesis

Hussain, W., Zowghi, D., Clear, T., Macdonnel, S., & Blincoe, K., (2016). Managing Requirements Change the Informal Way:When Saying 'No' is Not an Option. *24th IEEE International Requirements Engineering Conference,* Beijing China Sept 12-16, 2016.

Hussain, W., Blincoe, K., 2016. Establishing Trust and Relationships through Video Conferencing in Virtual Collaborations: An Experience Report on a Global Software Engineering Course. *Inaugural Workshop on Global Software Engineering Education (GSE-Ed'16) Part of 11th IEEE International Conference on Global Software Engineering 11th IEEE International Conference on Global Software Engineering,* Orange County, California – USA August 2-5, 2016

Hussain, W., 2016. Reflections on Requirements Change Management In Global Software Development: A Multiple Case Study. Doctoral Symposium. *Part of 11th IEEE International Conference on Global Software Engineering 11th IEEE International Conference on Global Software Engineering,* Orange County, California – USA August 2-5, 2016 Peters, A. K., Hussain, W., Cajander, A., Clear, T., & Daniels, M. (2015). Preparing the Global Software Engineer. *IEEE 10th International Conference on Global Software Engineering (ICGSE)*, 2015 doi:10.1109/ICGSE.2015.20

Hussain, W., & Clear, T. (2014). Spreadsheets as collaborative technologies in global requirements change management.*IEEE. 9th International Conference on Global Software Engineering (ICGSE) 2014*

Hussain, W., Buchan, J., & Clear, T. (2014). Managing Requirements in Globally Distributed COTS Customization. *IEEE International Conference on Global Software Engineeering Workshops (ICGSEW)*, 2014 doi:10.1109/ICGSEW.2014.13

Hussain, W., & Clear, T. (2012). *GRCM:* A Model for Global Requirements Change Management. *2nd International Requirements Engineering Efficiency Workshop* (REEW 2012), Essen, Germany.

Clear, T., Hussain, W., & MacDonell, S. G. (2012). The Many Facets of Distance and Space: The Mobility of Actors in Globally Distributed Project Teams. *IEEE Seventh International Conference on Global Software Engineering (ICGSE), 2012* doi:10.1109/ICGSE.2012.14

1.8 Thesis Structure

This thesis is organized in seven chapters (See Figure 1.1). Chapter 1 introduces the research work carried out for this thesis and provides background information about the study conducted for this PhD. In Chapter 2 the research questions that arise in the context and synthesis of the aim and objectives of this study are provided along with the review of relevant literature.

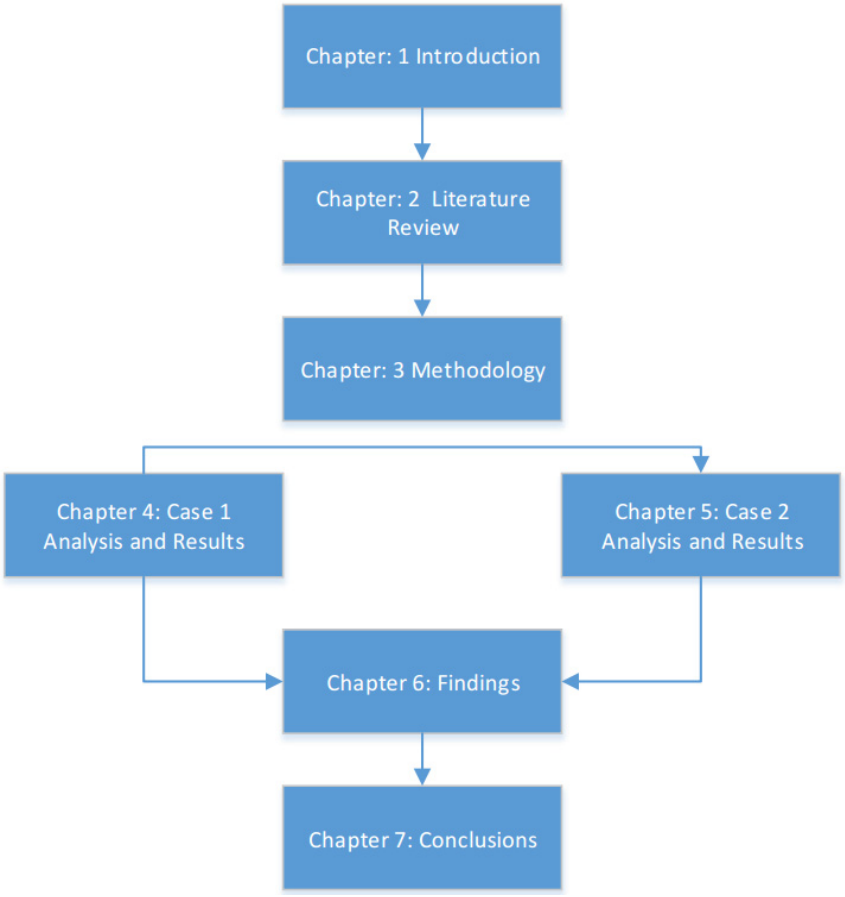

Figure 1. 1 Thesis Structure

The literature review is focused on the main areas of RM and change management in global software development and the tools and technologies applied to facilitate management of requirements. The detailed review highlights the paucity of research in the area of requirements change management in global software development and identifies the gap in GSD literature related to managing lifecycle activities of a requirements change.

In Chapter 3 the research methodology and design is presented along with a review of the guiding principles for the selection of a specific research method. It provides the rationale for using an interpretivist approach for this research work and outlines the multiple case

study exploratory methodology to carry out the research along with the data collection and analysis techniques.

In Chapter 4 and Chapter 5, the two studied cases are first discussed in detail based on their background and GSD contexts and then analysed by means of their prescribed vs. practiced models for requirements change management using lifecycle perspective of a requirements change in GSD projects. The chapters present the overarching GSD issues faced by the practitioners in each studied case and how these issues were exacerbated due to the geographical, temporal, socio-cultural and organizational differences between the organizations involved in each case.

In Chapter 6 the findings from these studied cases are compared, contrasted and discussed in light of the existing literature and theories about the challenges of RM in the GSD environment. Finally, in Chapter 7, a retrospection of the study is provided along with a discussion that outlines the contributions made by the study. An evaluation of this research work is also provided by discussing its limitations and the implications for research and practice are provided in this chapter. The aspects not covered in these seven chapters are included in the appendices.

"To change is difficult. Not to change is fatal." Ed Allen

"You don't have to change, survival is not compulsory" Edward Deming

Chapter 2. Literature Review

This chapter first provides the background of research in Section 2.1, by presenting a brief but critical review of basic concepts involved in Requirements Change Management in Global Software Development (GSD). Then in Section 2.2, it reviews the persistent challenges faced by practitioners in managing change related activities in GSD. Next in Section 2.3 the role of collaborative technology and the differing views on its usefulness to support change management activities are covered. Finally, in Section 2.4 the reasons to study this area are provided. The chapter ends with a summary of the overall topics discussed in this chapter in Section 2.5.

2.1 Background

2.1.1 Global Software Development

Developing software is a complex and interlinked practice where the people involved have to rely heavily upon each other's work. As a norm, software development today is organized as collaboration between geographically distributed individuals who speak different languages, belong to different socio-cultural backgrounds and are separated by different time-zones (Bjorn, Bardram, Avram, Bannon, Boden, Redmiles, De Souza, & Wulf, 2014 a). In fact, internationalization of software development and team distribution has increased tremendously in the past two decades (Bass, McDermott, & Lalchandani, 2015b). The landscape described here is considered in this thesis as GSD; also known as Global Software Engineering (GSE) (Smite, Wohlin, Feldt, & Gorschek, 2008). More precisely it can be defined as

"practices where geographically distributed collaborators are mutually engaged and thus interdependent in [both software engineering and] software development activities with the aim of designing, programming, and implementing an IT-system" (Bjorn et al., 2014 a).

Pioneered by software intensive high technology businesses and driven by enhanced network and communication technologies, the globalization trend has triggered changes in software procurement and software engineering processes (Bass, McDermott, & Lalchandani, 2015a). Global software development involves collaborative practices that might be organized as outsourcing (collaboration between different organizations); as

offshoring (collaboration within the same organization) or a mix of both (Bjorn et al., 2014). The next section (2.1.2 Requirements Change) presents requirements change as a reality in software development, explains the reasons of frequent change and discusses how it is generally viewed as a negative activity in traditional software development.

2.1.2 Requirements Change [4]

In the world of software, changes in requirements is an inescapable reality (Ambler & Lines, 2012; Jones, 1996; Sommerville, 2005). Application requirements are rarely stable or fixed making system evolution a fact of life in industrial environments. As rightly pointed out by (Weber & Weisbrod, 2002) *"Change and discussions about change are part of daily project life—and there's no way to change that."* Generally, 20% of requirements are expected to change in a software project however, in certain domains, 50% of requirements change is typical between critical design review and final entry into service (Nolan, Abrahao, Clements, & Pickard, 2011). In extreme cases up to 90% of requirements can change if requirements engineering activities are terminated prematurely (Berry, Czarnecki, Antkiewicz, & AbdElRazik, 2010) and (Verner & Abdullah, 2012).

There may be several reasons for requirements change. Ambler (2014) suggests that changes in requirements arise mainly because people change their mind on a regular basis. It happens when stakeholders realize that a) they have missed a requirement needed in the system, b) identified a bug and that has now become a requirement c) the actual needs were not understood initially d) the political landscape has changed giving rise to new priorities that motivate new requirements e) the place has changed d) the new legislations had to be adhered to either by adding new requirements or making changes to the existing system (Ambler, 2014). According to (Ebert, 2011) requirements are almost always uncertain before the development starts.

Stakeholders often face the dilemma of "I'll know it when I see it" because they find it hard to imagine the product or the final solution and state their requirements. Quite often

[4] For the purpose of this thesis, r*equirements change* mean the additions, deletions, and modifications to requirements (Costello & Liu, 1995) and it includes changes that result "in extensions to and alterations of the software's functionality and scope." (Carter, Antón, Dagnino, & Williams, 2001). Furthermore, misunderstandings due to inconsistent and vague requirements detected during the project are also considered a requirements change. (Ebert 2012).

their opinions about their stated needs evolve during elicitation and elaboration or the circumstances change leading to the changes in user requirements. This uncertainty caused by the cognitive limitations, which as per Ebert (2011), gives rise to requirements volatility. Similar views have been shared by Davis and Nori (2007), who suggests that requirements change because a) customers or users do not understand their own problem or opportunity, b) are unable to perfectly communicate their problems or needs to the system developers or c) system developers may have wrong perceptions of problems or opportunity. Looking from a broader organizational perspective Rolland, Salinesi, and Etien (2004) suggest that requirements change in response to the dynamic and competitive business environment where organizations themselves have to undergo frequent changes in order to survive or gain competitive advantage. This implies changes to their software based systems. Similarly, from a systems perspective, it is understandable that successful software tends to evolve because of the changes in the environment it operates in and also because the stakeholder requirements continue to change (Nuseibeh & Easterbrook, 2000).

In literature, several other factors have been identified that influence or trigger changes in requirements. McGee and Greer (2012) have provided a classification of these internal and external uncertainty factors. In their review of literature, they have identified thirty change triggers (such as changes to market demands, change to business case, increased customer understanding, resolution of incorrect or ambiguous requirements and so on) which are then classified under five domains; *Market, Organization, Vision, Specification and Solution*.

Changes in requirements, however are generally viewed as a negative activity, something that should be controlled if not avoided altogether (Davis et al 2008). They have been called one of software development's most 'chronic problems'- with no quick and perfectly effective cure (Jones, 1996). While there is some accommodation of the idea of changing requirements in the traditional as well as agile methodologies, both consider uncontrolled change as a risk to cost and quality of the software projects (McGee & Greer, 2012; Davis, 2013; Leffingwell & Widrig, 2003). This is perhaps due to a number of cases that have been reported that found requirements change at the heart of project failures (Standish Group 1999, 2004, Hall et al. 2002 cited by Somerville 2005). Davis et al., (2008) affirm that numerous studies have shown that the increase in change frequency escalates project costs, defect density and schedule (Davis, Nurmuliani, Park, & Zowghi, 2008). Recently, Berry et al., (2010) shared their experiences from a consulting

engagement with a client to help improve their Requirements Engineering (RE) process. The following quotation from one of the consultants interviewing the client explains how frequently changing requirements were viewed by the practitioners.

*"The Change Request process is perceived very negatively. Raising a Change Request is perceived as negative because it implies that somebody did not do his or her job … properly. Consequently, some system changes are performed in stealth mode, which avoids raising Change Requests. In this mode, changes are incorporated into the existing and compatible current work without creating a Change Request, code is up- dated, and the frozen requirements are updated silently, or code is updated without updating the requirements." (*Berry et al., 2010)

However, some studies rightly point to the need for showing flexibility and acceptance towards the possibilities of requirements change (Davis et al., 2008; Ambler 2012, p.30), requirements sufficiency (just enough requirements) (Davis, 2013) and reasonable correctness of requirements (Shaw, 2002). In such dynamically changing environments, applying rigidity in development approach and failing to recognize the need for evolution can make the system brittle (Cheng & Atlee, 2007) and actually drive additional costs (Nolan et al., 2011).

Today, there is an increased realization of software development in the changing world that requirements often emerge with system use rather than being pre-specified (Ruhe & Wohlin, 2014, p. vi). The less viable "sequential, top-down, requirements-first, reductionist "waterfall" approaches to manage projects and requirements are being forsaken to avoid problems such as delivering systems of little use to the users" (as their needs changed) (Ruhe & Wohlin, 2014, p. vi) and avoiding brittle systems and rigid architectures that fail at the first encounter of adverse conditions. Similar views are shared by Davis, Nurmuliani, Park, & Zowghi (2008), who suggest that if changes in requirements are based on stakeholders' real needs then avoiding or suppressing such needs is not a good option. It may be a short term strategy to increase the possibility of project success but puts product success at risk.

Thus, agile methodologies try to avoid the classical cliché of 'requirements freeze" and engage in software development with the vision to welcome consistent change (Dingsøyr, Nerur, Balijepally, & Moe, 2012; Turk, France, & Rumpe, 2014). Requirements change is seen as a means for adding value for the client, improving quality of deliverables, and

enabling satisfactory project outcomes. (Ambler, 2014; Gorschek, Gomes, Pettersson, & Torkar, 2012, p. 17; McGee & Greer, 2012).

In fact, some agile methodologies such as Extreme Programming (XP) and Disciplined Agile Delivery (DAD) encourage and accommodate changes in requirements during any stage of the development process even during an iteration (Ambler, 2014). This vision to accommodate change however, is not perceived as a simple submission to or an unwanted consequence of the inexorable laws of software evolution. Rather, it is embraced as the essence of software engineering (Cleland-Huang, Gotel, & Zisman, 2012, p. vi).

2.1.3 Requirements (Change) Management

The triggers of requirements change are constantly at work and propel software evolution (both development and maintenance) (Lehman, 1998; McGee & Greer, 2010). Software requirements change frequently at any time of the development process and managing change; a fundamental RE activity, is crucial to ensure high quality which is necessary to meet the end user needs of its users (Bohner, 1996; Lam, Loomes, & Shankararaman, 1999; Tanabe, Uno, Akemine, Yoshikawa, Kaiya, & Saeki, 2008). It is widely accepted that successful requirements engineering encompasses managing requirements evolution (Cheng & Atlee, 2007; Davis, 2013; Sommerville, 2011; Wiegers & Beatty, 2013). The activities involved in RCM are aimed at developing a shared, documented understanding of the requirements to enforce a mechanism that controls volatility so that the system could satisfactorily fulfil requirements at the time of delivery (Lang & Duggan, 2001).

Due to the size and increasing complexity of modern day software, requirements and change management has become one of the most important yet most complex parts of the requirements engineering process (Damian, Chisan, Vaidyanathasamy, & Pal, 2005; Hull, Jackson, & Dick, 2010; Raatikainen, Männistö, Tommila, & Valkonen, 2011). People involved in GSD continue to see requirements change management as the ‘most difficult part’ of software development. Similarly, practitioners responsible for managing changes in requirements are having to wrestle with these challenges on a daily basis (Hussain & Clear, 2012).

According to Brooks (1987) the iterative extraction and refinement of requirements (that spawns the lifecycle of a project) is the single hardest part of software development. Getting the requirements wrong has the most crippling effect on software systems and its late rectification is more difficult than any other aspect of software development (Brooks Jr, 1987) as evident in for example (Berry et al., 2010) and (Nolan et al., 2011; Seth, Mustonen-Ollila, Taipale, & Smolander, 2012; Verner & Abdullah, 2012).

This thesis uses a broader definition of RCM where it is seen as part of RE that lies at the core of software engineering and is concerned with all processes involved in changing system requirements (Sommerville & Sawyer, 1997; Torkar, Gorschek, Feldt, Svahnberg, Raja, & Kamran, 2012). It is not seen as an activity only related to managing change documentation once the specifications are finalized as noted in Chapter 1, pg. 1). Rather it extends to include development activities leading to change implementation, testing as well as support activities (such as traceability and configuration management). Davis (2013) presents a similar view of requirements management and considers it as the first activity to be initiated in the project[5], and one that spawns the complete software development life cycle (Davis, 2013, pp. 6-7). It includes requirements elicitation *triage* (analysis, negotiation & prioritization), specification, traceability and managing changes in requirements over time (Davis, 2013, pp. 6-7; Torkar et al., 2012).

Requirements (and change) management is considered to have a significant impact on both bespoke and market driven software development success (Gorschek, Gomes, Pettersson, & Torkar, 2012). The direct but negative impacts of poor requirements change management relate to reduced product and service quality, unsatisfactory technical and commercial outcomes, delays and non-fulfilment of original expectations and even project failure as covered in many studies (Dalcher, 2014; Ebert, 2011; Sommerville & Sawyer, 1997). It is therefore imperative to carefully manage requirements on an on-going basis because they have a tendency to quietly alter the context of software problems in fundamental ways (Damian et al., 2005; Lang & Duggan, 2001; Tanabe et al., 2008). On the other hand, having an adequate process to manage requirements and change related activities is considered crucial to the successful delivery of a project (Gorschek et al., 2012).

Effective management of changing requirements is reported to result in effective project negotiations, reduced requirements creep and accurate effort estimation (Damian & Chisan, 2006). In GSD effective management of requirements is considered to be the

[5] This treatment is in line with the views of software requirements management working group which considers that *"Changes to requirements and mission needs are inputs to Elicitation. These changes are transformed through this activity to candidate technical requirements, as well as non-technical requirements, and after exiting from this activity are no longer considered changes"* (RMW Report, 1998, p.12)

ultimate success factor (Ebert, 2011, p. 39). Some authors go to the extent of suggesting that the countries that can devise the best way to perform RM are likely to become the IT leaders of the future (Davis & Hickey, 2009, p. 161). Many companies therefore, devote substantial resources to reactive development; which means reacting to requirements coming in today, to survive the constant bombardment of requirements from external sources (Gorschek, Fricker, Palm, & Kunsman, 2010). Similarly, organizations with a low maturity are focusing on requirements management as a key area for their process improvement endeavours (Hurtado Alegría, Bastarrica, Quispe, & Ochoa, 2014; Niazi, Hickman, Ahmad, & Ali Babar, 2008).

Ebert & De Man (2005) posit that a common denominator of changes in requirements is project delay which is almost always the likely outcome. They propose key techniques to deal with continous requirements change which include evolutionary lifecycle prototyping, reduced cycle time from inception to delivery, sensitivity analysis (i.e., determining the localization, scope and impact of changes) and practical risk management (e.g. traceability, improved maintainability and isolating features that are likely to change). They contend that except for iterative development, none of these techniques are really applied in mainstream usage and neither is their combination used in practice. As a result we continually witness cases where poor management of requirements negatively affects project outcomes. Two such recent studies are reported here;

Using an exploratory case study on a large scale and high profile IT outsourcing project (BSkyB), which was subject to recent litigation in the British High Court, Verner & Abdullah (2012) identified six factors for the failure of this project. Out of these six, the two main factors were directly related to poor requirements management: *scope and requirements*, *planning and control*. The authors report that from the start, unclear and inadequate requirements made it difficult to manage the project and handle ambiguities unfolding due to the ill-defined requirements. Furthermore, the reports from some of the key stakeholders suggested that requirements kept emerging like “handkerchiefs from a magician’s sleeve”, as much as three years into the project (Verner & Abdullah, 2012). The client analyst termed the project as a classic case of “scope creep” because of continuous changes in requirements which made it difficult to define the project specification to set a requirements baseline and get a good grip on managing requirements. The vendor attributed the undefined and constantly changing requirements to be the cause of project and cost overruns (Verner & Abdullah, 2012).

Similarly, Seth et al. (2012) conducted a qualitative case study on identifying practices that enable quality construction in software development. The findings suggest that requirements elicitation is an ongoing process which continues even beyond product delivery; a clear indication of a continuous onslaught of requirements change. The following quotation by a developer from their study suggests that practitioners responsible for managing changes in requirements in GSD are wrestling with these challenges on a daily basis as noted by (Hussain & Clear, 2012). It further emphasises the fact that practitioners continue to see RCM as the 'most difficult part' of software development.

"There are always problems when the changes occur. Requirements change management is always the most difficult part. So in the beginning of the project we have agreed that this is the scope…then, usually the requirements change and then we have to manage it. We get a change request and then we have to analyze it and then we have to decide how long it takes and then negotiate with the customer that are they willing to pay for it or just go with the contract or whatever." Tester and Developer in one of the reported cases by (Seth et al., 2012).

The next section (2.1.4) reviews the literature on RCM presented as a challenge in GSD. The next section (2.2) classifies persistent RE and RCM challenges reported in various GSD studies in four different categories and discusses them under each relevant category.

2.1.4 Requirements Change Management in GSD- a Challenge

In the GSD landscape, new outsourcing business models compel requirements engineering and change management activities to be increasingly performed across distributed locations, organizations, participants, and time (Damian, 2007; Hansen, Kharabe, & Lyytinen, 2013; Šmite, Wohlin, Galviņa, & Prikladnicki, 2014). Basic management of requirements from distributed stakeholders within complex domains is still a challenge (Nolan et al., 2011; Raatikainen, Mannisto, Tommila, & Valkonen, 2011). GSD adds further challenges for the collaboration-intensive activities of requirements and change management where stakeholders need to specify and manage requirements across geographical, temporal and cultural distance (Damian, 2007; Laurent, 2010; Schmid, 2014). Numerous studies over the years have made it clear that RM in GSD is still an arduous task (Hussain & Clear, 2014; Lai & Ali, 2013; Prikladnicki, Audy, & Evaristo, 2003; Sinha, Sengupta, & Chandra, 2006; Šmite, 2006). Researchers have identified specific GSD challenges and (some of the) solutions to carry out requirements engineering and requirements management activities. These however are

scattered (and reported) in numerous research studies (Bhat, Gupta, & Murthy, 2006; Cheng & Atlee, 2007; Damian, 2007; Damian & Zowghi, 2003b; Damian & Zowghi, 2002; Ebert & De Neve, 2001; Hanisch & Corbitt, 2007; Lopez, Nicolas, & Tov, 2009; Nolan et al., 2011; Nurdiani, Jabangwe, Smite, & Damian, 2011; Prikladnicki, Audy, & Evaristo, 2003; Mikko Raatikainen et al., 2011; Schmid, 2014; Šmite, 2006) (A collated list of those RE related challenges is provided in appendix A).

Recognition of Requirements Change Management (RCM) as a challenge is not new for GSD as it is known to pose challenges since the inception of this paradigm (Schmid 2014; Herbsleb & Grinter, 1999). Managing customer requirements through the use of concurrent engineering utilizing geographically distributed development groups was reported as a problem even in the mid-nineties (Rafii & Perkins, 1995). Similarly, in the early days of GSD, Herbsleb and Grinter (1999) in their seminal work reported challenges in managing changes in requirements due to a higher degree of coordination overhead and unanticipated but frequent coordination breakdowns that led to delay, inefficiency and frustration (Herbsleb & Grinter, 1999). Here we are, almost two decades later, still facing the distance-exacerbated challenges in nearly all of RM associated activities namely elicitation, negotiation, analysis, prioritization and traceability are considered to be more challenging in GSD (Calefato, Damian, & Lanubile, 2012; Cleland-Huang, Chang, & Christensen, 2003; Parviainen & Tihinen, 2011). The question is, why is carrying out RCM activities in GSD more challenging? The answer is not straight forward as it involves several (distance related) factors (Noll, Beecham, & Richardson, 2010). The review of existing literature can perhaps help us in understanding some of the causes of these challenges.

In this thesis, I use four major GSD problems as broad level categories to discuss already reported but persistent (somewhat addressed but yet to be resolved) challenges of RE and RCM activities in GSD. The categories are adapted from the model proposed by Damian and Zowhghi (2003 a) in their seminal work on RE in GSD see Figure 2.1. These categories include *Inadequate Communication, Cultural Diversity, Knowledge Management and Time Difference*. Damian and Zowghi (2003a) reported that these known and major (high level) GSD challenges adversely impact eight requirements engineering activities namely, *Elicitation, Analysis, Negotiation, Validation, Prioritization, Examining Current System, Specification, Managing Uncertainty*. Using these four main problems as broad level categories this thesis tries to collate and classify

various RE and RCM challenges reported in numerous GSD studies, under each relevant category reported.

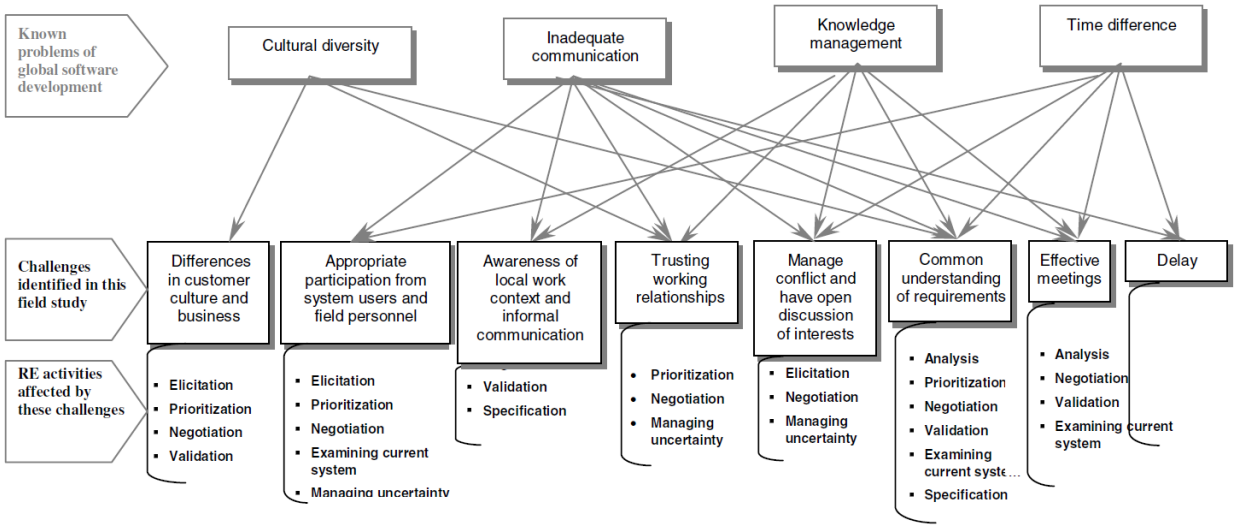

Figure 2. 1 A model of impact of challenges and the affected RE activities in GSD by (Damian & Zowghi, 2003b)

2.2 Persistent Challenges in Managing Requirements in GSD

In GSD the fundamental problems of coordination and control are exacerbated because of the negative impact of distance on communication (Noll & Beecham 2010; Carmel & Agarwal, 2001). The distance dimensions (geographical, temporal and socio cultural) hinder almost all of the activities involved in requirements management (de Gea, Nicolás, Alemán, Toval, Vizcaíno, & Ebert, 2013; Herbsleb, 2007). The coming sub sections deal with various persistent GSD challenges related to RE and RCM. These challenges are discussed under specific categories namely; *Communication Challenges, Knowledge Management Challenges, Cultural & Organizational Diversity Challenges* and *Time Difference Challenges* by (Damian & Zowghi, 2003). These challenges have significant impact on requirements and change management activities discussed in detail in the coming subsections.

2.2.1 Communication Challenges

Cross-site communication is often attributed as the single biggest source of difficulties in outsourcing and offshoring projects that hinders coordination and causes most of the challenges (Ebert, 2014). Communication is also considered to be the most affected area of software development that suffers due to geographical, temporal and language differences (Hanisch & Corbitt, 2007). In GSD, limited opportunities to hold face to face meetings and informal communication adversely affect work coordination and control (Herbsleb, 2007). This has a direct impact on RCM activities that are people-centric and

collaboration intensive (Hanisch & Corbitt, 2007; Herbsleb 2007, Damian, 2007; Damian & Zowghi, 2003). The coming subsections deal with the two main areas (*Requirements and Change Awareness* and *Shared Understanding of Requirements and Change)* related to change management that are affected by inefficient or inadequate communication due to distance.

2.2.1.1 Requirements and Change Awareness

The sharing of joint awareness allows two people to establish common grounds (knowledge that is common among participants) (Olson and Olson 2000) which is necessary to carry out any form of joint activities. For collaborative processes such as software development the inherent needs of requirements and change related awareness are particularly important (Dourish 1992). The main barrier to collaboration is the awareness of the task on hand and the information to complete that task successfully (Damian, Chisan, Allen, & Corrie, 2003). For effective team communication and collaboration, a vast amount of information or mutual knowledge sharing is a must (Neale, Carroll, & Rosson, 2004). In GSD, awareness makes the distributed requirements management activities possible and helps in ensuring that individual contributions remain relevant to the whole group (Lanubile, Calefato, & Ebert, 2013). Several studies in GSD point out the accentuated awareness need for cross site project, task, requirements, and change related activities (Lanubile et al., 2013; Omoronyia, Ferguson, Roper, & Wood, 2010; Sarma & Van Der Hoek, 2006).

Catering for awareness needs in distributed settings is however difficult due to the constrained channels for informal and face to face communication (Damian et al., 2003; Lanubile et al., 2013; Omoronyia et al., 2010; Šmite, 2006). Omoronyia et al (2010) report that achieving similar levels of awareness in distributed settings to those available in co-located settings is far more challenging because the stakeholders are separated by time and space. They argue that awareness is highly contextual and therefore is not generalizable across individuals, entities or activities found in distributed collaboration. The contextual nature of this awareness comes from that fact that, over time and while working on different tasks, people form different perceptions about their colleagues, their role, the work they perform and their workspace. According to Omoronyia et al. (2010) this type of c*ontext awareness* is essential but needs to be supported more for successful distributed collaborative software development. Neale et al. (2004) advocate that establishing *activity awareness* support for distributed collaborators is a core challenge

for CSCW systems. In order to support the change management process, team members in GSD need to be aware of the tools and resources available to their counterparts, their knowledge and expectations, current focus and action and the criteria they will use to evaluate joint outcomes. Such activity and contextual background information is significantly fractured in distributed development environments often due to lack of communication. The authors argue such information is often removed from immediate interaction and for the most part totally unsupported within the existing technologies. Furthermore, due to lack of communication, team members make assumptions regarding the contextual and background information, which result in breakdowns that are frustrating to users.

According to Bjarnason et al., (2011) requirements change related awareness has to be catered for at all the stages of the software development process. They consider that lack of continuity in requirements awareness, often due to inadequate communication, during the lifecycle of the project causes gaps in vital requirements knowledge which can lead to misunderstood and incorrectly implemented requirements, and poor decision making (de Gea, Nicolás, Alemán, Toval, Vizcaíno, & Ebert, 2013). Similarly, the authors emphasise the need for communicating change related essential contextual information among stakeholders to aid requirements stability and (common) understanding.

In GSD when changes in requirements happen, the prescribed and formal apparatus to disseminate change related information (such as a requirement specification document) is slow to react. The untimely and slow propagation of change related information often results in limited awareness, visibility and control of remote resources as well as lack of shared context awareness (Heindl, Reinisch, & Biffl, 2007b; Herbsleb, Mockus, Finholt, & Grinter, 2001). It also results in implementation of outdated requirements and wasted effort by the developers who have to do re-work on the code (Heindl, Reinisch, & Biffl, 2007).

Kommeren and Parviainen (2007) highlight the lack of awareness about the tasks carried out by different team members as a major issue. It leads to problems such as: team coordination and managing requirements. Herbsleb (2007), suggests that such lack of *task awareness* makes it difficult to initiate contact often resulting in misunderstanding of the content being communicated (for example requirements). Ebert (2006), considers that further delays in carrying out change related tasks are experienced if all stakeholders are not notified instantly about the rationale for making a particular change. More importantly in case of requirements change, task unawareness critically impacts the ability of

distributed team members to track the effects of changes in distributed collaboration space making change related activities (management, implementation, testing, traceability) problematic (Herbsleb, 2007). There is also a possibility of wasted effort and re-work because some of the already discussed and seemingly resolved issues are rehashed due to limited collective memory or 'organizational amnesia' (Catledge and Potts, 1996; Damian, 2007).

2.2.1.2 Shared Understanding of Requirements and Change

Shared understanding of requirements between stakeholders and software engineers is essential for successful development of software systems (Glinz & Fricker, 2013). Shared understanding is established through stakeholder interaction during RE activities (both in traditional development and agile environments) by means of requirements related artefacts such as specification documents, user stories, and up-front test cases (Glinz & Fricker, 2014). Inadequate and inefficient communication due to geographical, temporal or cultural distance creates gaps in the shared understanding of requirements. (Schneider et al., 2013) suggest that achieving shared understanding of requirements is critical but challenging in GSD because of the diversity of stakeholders' professional background and interests. Schneider and colleagues (2013) consider that to enable shared requirements understanding between distributed stakeholders it is critical that the challenges both at the initial capturing and the subsequent handling of requirements (and changes) are dealt with (Schneider et al., 2013). Frequent face to face and informal communication is known to serve well and improve shared understanding of the requirements. However, in GSD where loss of context and loss of communication is quite common, achieving shared understanding is challenging (de Gea et al., 2013; Herbsleb, 2007).

Requirements management, a collaborative and knowledge intensive activity, is hindered if the stakeholders do not have a shared mental model of requirements (de Gea et al., 2013). Parviainen and Tihinen (2011), report in offshoring contractual agreements that it is not easy for the client and vendor team to hold intense communication and interact frequently to establish good change understanding. According to Parviainen and Tihinen (2011), lack of shared understanding results from the inability to communicate and share tacit knowledge (knowledge not yet put into words) during various requirements related activities such as gathering, prioritizing, and communicating requirements (and change) information (Ma, Nuseibeh, Piwek, De Roeck, & Willis, 2009). They consider levels of

common understanding of requirements and changes may drop due to inadequate levels of requirements description at the time of definition (due to limited time and resources). Ghaisas and Ajmeri (2013, p. 144), report that articulating and clarifying business problems in an understandable manner to arrive at a specification based on a shared understanding is challenging in a distributed development environment.

Frequent interactions, communication and knowledge sharing is needed to resolve certain conflicts as written documentation is often inadequate when resolving misunderstandings related to requirements or changes in requirement specifications (Damian & Zowghi, 2002; Agerfalk 2008). Ghaisas and Ajmeri (2013, p. 144) note that for distributed stakeholders, that have different backgrounds, developing and utilizing a common vocabulary, assigning meanings to various business concepts, determining their interrelations, and reconciling multiple viewpoints becomes very challenging. They further suggest specifying requirements and changes in a manner understandable to all relevant stakeholders. The authors argue that continually accommodating changes in the shared understanding is a challenge in GSD (Ghaisas & Ajmeri, 2013, p. 144). Similarly, Neale et al. (2004) suggest that altering shared context is the single most significant reason for the failure of adoption and use of distributed systems.

However, Herbsleb (2007), posits that, for any given project that vision for a successful global development rests on its capability to achieve shared understanding of requirements and effectively manage change. As per Damian (2007), achieving shared understanding of requirements is more complex in GSE; as it requires redefined interactions (more interactive ways of communication and coordination), fast relationship building and knowledge management. She further asserts that the processes that can help achieve shared understanding include a) knowledge acquiring and sharing processes that enable exploration of requirements, application domain and technical solutions b) iterative processes that enable shared understanding throughout the project and c) effective communication and coordination processes that support a) and b).

2.2.2 Knowledge Management Challenges

Knowledge is a major organizational resource especially for knowledge intensive activities such as software development and requirements engineering (Manteli, van den Hooff, Tang, & Van Vliet, 2011). For effective team communication and collaboration a vast amount of information or mutual knowledge sharing is a must (Neale et al., 2004). In requirements engineering, KM is seen as an umbrella activity comprising of a number of tasks related to management of requirements and their evolution over time and across

product families (Cheng & Atlee, 2007). Noble (2004), points out the central role played by knowledge in team collaboration which enables team members to avoid predictable errors caused by knowledge deficiency. Effective KM has been coined as an enabler to achieve the potential GSD benefits of reduced rework, increase productivity by repeatedly employing successful processes that can lead to increased quality, and better decision making (Rus & Lindvall, 2002). He posits that teamwork is effective when the members know what they need to know, and are aware of how to acquire knowledge when needed.

Knowledge sharing and acquisition in geographically distributed development environments is not easy because of the difference in stakeholders' background, language and culture that hinders in establishing trust. It is most evident during requirements management where lack of trust and the inability to share knowledge often results in an uncommon methodology, which hinders the exchange of critical information related to requirements (Damian, 2007). Boden, Avram, Bannon, and Wulf (2009), share this view and further add that managing knowledge in globally distributed teams is challenging due to "multiplicity of organizational, temporal, spatial, legal, national and cultural barriers" which may affect development pace and the quality of software produced. Boden, Avram, Bannon, and Wulf (2009) analyse knowledge sharing practices and illustrate how social issues influence knowledge sharing especially on maintaining awareness and the collaborative use of shared artefacts. Their findings suggest that "organizational culture is permanently re-negotiated and adjusted to fit the distributed collaboration, as the teams learn how to deal with each other". According to Parviainen and Tihinen (2011), knowledge-related challenges can complicate work in GSD projects where the inability to transfer critical information may cause a failure. Herbsleb (2007), considers that seeking and integrating necessary knowledge is more difficult in a global context. Sinha and Gupta (2006), while reporting the challenges faced by global software development practitioners note that managing requirements knowledge in any given project is challenging for distributed teams. When new team members join the project a considerable time has to be invested to learn about requirements and why certain changes to those requirements were made and how they could handle future changes.

Addressing knowledge-related challenges holds a central role in order to develop solutions that can support GSD (Maalej et al., 2008; Parviainen & Tihinen, 2011). Maalej et al. (2008) consider existing RE processes and tools do not adequately support requirements knowledge management which can improve participant collaboration in distributed projects, enable traceability and support requirements evolution (Dutoit,

McCall, Mistrík, & Paech, 2006; Dutoit & Paech, 2003). At present knowledge residing within information rich resources (repositories of project memories, version files, change histories, and other requirements related documents) is underutilized in GSD (Herbsleb, 2007; de Gea et al., 2013). Thus, achieving good KM through knowledge sharing to better coordinate, make decisions, achieve consensus and improve team effectiveness, is vitally important for successful GSD (de Gea et al., 2013).

2.2.3 Cultural & Organizational Diversity Challenges

Requirements engineering challenges are compounded by inherent cultural boundaries imposed on distributed projects and teams (Hsieh, 2006). Highly idealized collocated teams are fundamentally different from geographically distributed teams in several ways. In contrast to distributed teams, work coordination of collocated teams is supported by their history of working together, knowledge about team members' expertise and responsibilities, frequent interactions (formal and informal) (Herbsleb, 2007). They also share a common set of habits and vocabulary developed over time by working together. But most importantly they often share a common language and national as well as organizational culture (Herbsleb, 2007) - not the case of RM in GSD. Cultural issues during requirements management activities are compounded when distributed stakeholders belong to different cultural zones (like Asian vs. European) (Schmid, 2014).

For example, Brockman and Thaumuller (2009) discuss experiences of a German and Chinese collaboration over distributed agile requirements engineering activities and report several problems which arise because certain values of citizens from these countries are in direct conflict. For example, being open and direct in criticism (German team) vs indirect and more considerate (Chinese team). Due to the indirect approach, the German team felt that requirements were expressed ambiguously. Similarly, the Chinese team worded their feedback for unit testing in a polite way which was again misinterpreted as a positive validation of the unit tests conducted.

Noll et al. (2010), report that collaboration among people who come from various cultures and backgrounds with different languages makes understanding each other and working together challenging. Difference in language also plays an important role in requirements related communication. Since English is considered the lingua franca of GSD, the quality of communication is affected especially if the remote team members come from non-English speaking countries and have limited language skills. Lacking language skills, some team members resort to asynchronous mode of communication (e.g. emails) which can be an impediment to communication especially when video and teleconferencing

facilities are available and considered important by other team members (Noll et al., 2010). Similarly, Parviainen and Tihinen (2011) suggest that requirements communication suffers due to cultural and language differences when people do not understand the things in the same way and have to rely on a clear, consistent and unambiguous description of requirements. Herbsleb (2007) considers that carrying out requirements related negotiations and resolving conflicting assumptions in the face of process mismatches, differences in technical and domain vocabularies, and incompatible environments (as in the case of GSD) can be particularly problematic (Herbsleb, 2007). Bhat and Gupta (2006) report their experiences from the study of client-vendor relationships in real life offshore-outsourcing projects and (in agreement with the literature) identify cultural diversity as one of the main challenges to RE in such projects (Carmel, 1999).

Hanisch and Corbitt (2007) argue that to create lasting client relationship and successfully achieve RE activities, social and cultural aspects have to be considered. They report the case of cultural differences between distributed team members of United Kingdom and New Zealand teams that posed challenges for RE despite the strong relations between the countries and their cultures. Challenges such as invalid requirements, miscommunication and misinterpretation of requirements, difference in working culture between a governmental and commercial organization and the resulting communication overhead were among the main difficulties identified.

According to (Hsieh, 2006) the development of shared understanding of requirement also suffers due to the diversity in people's cultures. Team members in globally distributed projects often come from different cultures and often have drastically different values, beliefs, and disparate approaches to communication and problem-solving. They might also have contrasting interpretation of requirements and the ways to perform RE which hinders the development of a shared mental model for these team members (Hsieh, 2006). In GSD, this can lead to miscommunication and misinterpretation (Hsieh, 2006). George, Owoyemi, and Onakala (2012) performed a qualitative study using exploratory semi-structured interviews conducted on one hundred participants to understand the relationship of culture and the experiences they had in their organisations. They conclude that the 'unlearning' of the cultural aspects (traits, patterns of thinking, feeling and potential acting) are more difficult than the 'learning' processes; signifying that culture is enduring in people's mind. In the language of computers this cultural retention is

referred as ‘the software of the mind’ which according to Hofstede is difficult to ‘unlearn’ (Hofstede & Hofstede, 2001, p. 1).

In his case study of two GSD projects Prikladnicki (2003) reported cultural differences and trust among the most influential factors that make requirements management critical in GSD. Their findings suggest that cultural and trust related problems can be addressed by training distributed teams in soft skills such as awareness about cultural differences and establishing trust so that they can easily communicate and exchange knowledge. They emphasise that knowing about the background and culture of the people you work with is important to establish trust and enable mutual consideration about the manner in which you communicate.

Similarly, Smite (2006) carried out an exploratory research study to understand the practices of RM carried out in GSD and to identify threats to managing requirements. Her results show that one of the top 5 threats to requirements management in GSD is the organizational and cultural difference which often leads to diversity in process maturity and/or inconsistency in work practices and culture mismatch between the partners. She further reports lack of language skills and terminology differences are among the additional threats and reasons for poor RM in GSD. She also includes ‘poor cultural fit’ as one of the risks that can put requirements clarification under threat (Smite 2006).

Several approaches are noted in GSD literature to address cultural issues such as a) choosing sites which are culturally similar such as America and Ireland (Carmel & Agarwal, 2001), b) using a ‘cultural ambassador” or “liaisons” who understands the culture and language for both client and the developers (Deshpande, Richardson, Casey, & Beecham, 2010; Lings, Lundell, Ågerfalk, & Fitzgerald, 2007) c) improving cultural awareness by having site visits or video-conferencing (Noll et al., 2010 d) rotating managers to different site to gain first-hand knowledge of how various sites work (Lings et al., 2007)

2.2.4 Time Difference Challenges

In GSD, time zone difference between sites can be very influential and often dictates the timing of when (team as well as individual) interactions occur and when tasks are processed (Espinosa & Carmel, 2004). Geographically distributed teams often encounter challenges due to time zone differences (Espinosa & Carmel, 2004). Noll et al. (2010) report that temporal distance is seen as a barrier to collaboration. Few overlapping hours due to time zone differences make coordinating same-time (i.e. synchronous) meetings with distributed stakeholders difficult (Noll et al., 2010). Delay in responding to

asynchronous communication is often cited as a problem resulting from temporal distance (Carmel, 2012). For example, Herbsleb and colleagues (2001) report a two–and–a–half time's mean delay in receiving response to some enquiries. Similarly, in another study they report that implementing change requests took twice as long when carried out in a distributed context (Herbsleb & Mockus, 2003). In GSD literature however, distance and time separation are often correlated or even mixed making it difficult to tease out the real effects of time zone difference (Espinosa, Cummings, Wilson, & Pearce, 2003). For example, the study carried out by their study that involved distributed software teams recognize that the observed mutual knowledge problems are not caused by distance alone rather a combination of factors such as technology used, cultural differences and time zone differences (Cramton, 2001).

There are strategies to mitigate issues arising from time zone differences such as "follow the sun" or "round the clock" software development that try to offset some of the problems of distance separation and speed up project work (Carmel, Espinosa, & Dubinsky, 2010). However, such approaches have not yet overcome the challenges of temporal differences and continue to suffer from managing daily handoff cycles or handing off work-in progress. (Kroll, Hashmi, Richardson, & Audy, 2013). So, temporal distance is still a challenge in GSD introducing delaying in completion of distributed development tasks and getting feedback to work (Nurdiani et al., 2011).

The next section 2.3 provides a brief overview of some other challenges related to RCM in GSD.

2.2.5 Other Challenges

2.2.5.1 Requirements Volatility

Initial planning and the original business case is undermined by frequently changing requirements and when these changes are improperly or insufficiently managed they lead to inconsistencies and defects (Ebert 2012). Similarly, Kraut and Streeter (1995) enumerate several factors for the inherent uncertainty in software development such as business needs, user desired, computer platforms, input data and the physical world itself for which the software was designed, as well as deficiencies in requirements specifications which lead to requirements volatility. In their literature review Schneider et al. (2013) report that for distributed teams' continuous tracking of changes in requirements is a challenging task. Similarly, (Herbsleb & Mockus, 2003) highlighted that GSD projects have to deal with delay as one of the main challenges due to constantly

changing requirements. They contend that delay often results due to late propagation of requirements change across sites caused by slow reacting formal mechanisms that communicate change (such as software requirements specifications). Similarly, Nurmuliani and colleagues (2004) consider management of requirements volatility as a multifaceted problem which is a challenge for GSD. Changes in requirements, especially in the later phases of the development project required additional communication between the stakeholders which led to communication overhead and project challenges (Hanisch & Corbitt, 2007). Herbsleb (2007) argues that "getting the requirements right, and dealing with unstable requirements" is a common challenge that needs to be addressed and as part of the vision for successful GSD. Dealing with software changes in distributed settings remain chronic coordination problem, where geographical, organizational or social barriers reduce opportunities and eagerness of people to share information and learn from distant colleagues (Newcomb, 1961) cited by (Kraut, 1995).

2.2.5.2 Traceability

According to Ramesh and Jarke (2001) RT is aimed at ensuring continued alignment between stakeholder requirements and system evolution (Ramesh & Jarke, 2001). Traceability is used to describe and follow the life of any conceptual or physical artefact and facilitates information sharing (and awareness) especially for change impact analysis both during software development and maintenance (Ramesh & Jarke, 2001; Parviainen & Tihinen, 2011; Wiegers, 2003). Traceability helps to understand the relationships within and across software artefacts such as requirements, design and code (Wiegers, 2003). One of the key activities of requirements management is to assure requirements traceability throughout the artefact's lifecycle (Torkar et al., 2012), yet GSD is known to exacerbate the challenges of performing all activities related to traceability (Buchan, Ekadharmawan, & MacDonell, 2009; Cleland-Huang et al., 2003). Similarly, Tihinen, Parviainen, Suomalainen, and Eskeli (2015) report that a lack of traceability is one of the challenges in GSD projects. Teams working in GSD environments face additional challenges in creating and maintaining the traceability information because such information resides on and shared through various tools and processes over geographically distributed company borders (Tihinen et al., 2015).

2.2.5.3 Software Configuration Management

The management of a software system as it evolves from a bare concept to a software product that can be used with confidence is termed as software configuration management (SCM) (Berlack, 1992). As a discipline, SCM involves controlling the evolution of complex software systems and their components so that their correctness and consistency can be ensured (Mohan, Xu, Cao, & Ramesh, 2008). For change request implementation, SCM helps to identify the relevant change, understand the impact of making changes, and facilitates the tracking of changes (Estublier, Leblang, Hoek, Conradi, Clemm, Tichy, & Wiborg-Weber, 2005; Hass, 2003). The findings from a systematic literature review by Fauzi et al., (2010) suggest that the lack of coordination and group awareness in globally distributed projects result in additional challenges related to SCM. They argue that although SCM issues do arise in collocated environments as well, however they are exacerbated in GSD due to the differences in processes, tools employed at different sites. Furthermore, they suggest that solutions used for SCM problems in collocated environments do not apply or work in GSD.

Several other challenges have been identified in other studies regarding the difficulties of performing SCM in GSD. Mishra and Alok (2011) report that managing consistency and concurrency of project artefacts in GSD environments is among the major issues for SCM. Shrivastava (2010), report that the networks spanning globally distributed sites are often unreliable and slow making SCM difficult. They suggest that tasks such as critical data transmission are challenging and have to be meticulously planned and executed (Shrivastava, 2010). Parviainen and Tihinen (2011), studied knowledge management practices in distributed development environments and report that tool selection for SCM (knowledge intensive support practice that enables collaboration) was one of the main challenges for distributed team members (Parviainen & Tihinen, 2011). They report that agreeing about the configuration management tools was difficult as it raised a great deal of sentimental arguments by people with different backgrounds and preferences regarding the tools of their choice. Since SCM is strongly related to requirements change all of these activities affect requirements change management.

Having covered the persistent challenges of GSD reported in the literature this research sets out to identify and report contemporary challenges to managing requirements change based on the empirical data from this multiple case study. The specific resaerach question related to RCM challenges in GSD which this study attemps to answer is noted at the start of Chapter 3.

The next section covers the support offered by collaborative technologies to RCM activities. The literature review on CTs presented here also covers the successful usage of CTs, their limitations and the continued need to understand the role of CTs in globally distributed einvironments for managing RCM.

2.3 Role of Collaborative Technologies in GSD and Requirements Management

The review of literature carried out to understand the role of existing collaborative technologies in GSD for this thesis highlights the differing views regarding CTs and their use. Researchers in general agree that collaborative tools and technologies are necessary and helpful to carry out global software development activities. However, a deeper study of the literature brings out the myriad of experiences and opinions where some suggest that the existing CTs are adequate while others consider the need for improvements in the technologies. They report the deficiencies of these technologies, issues in their usage, challenges in the implementation of suitable combinations of technologies to support GSD work. In the coming subsections first an overview of the available collaborative technologies is provided and then the differing views about CTs are briefly discussed.

2.3.1 Collaborative Technologies – a Necessity in GSD and RCM

Software engineering is inherently a collaborative activity that requires people to work together and achieve desired results together (i.e. develop better software) (Whitehead, Mistrík, Grundy, & van der Hoek, 2010). Collaboration over requirements entails a complex interplay of cognitive and behavioural processes that are required for idea generation, decision making and requirements conflict resolutions (Nuseibeh and Easterbrook 2000). Distance (spatial, temporal and socio technical) places many barriers into the pathways of collaboration thus exacerbating the existing challenges of requirements related collaboration (Lanubile, Ebert, Prikladnicki, & Vizcaíno, 2010; Noll et al., 2010; Whitehead et al., 2010). Globally distributed development teams often suffer from insufficient collaboration (Lanubile et al., 2013) and often try to deal with distance by using some communication (and collaboration) media (Zowghi 2003). GSD, thus highlights a pronounced need for improved collaboration facilitated by tool and collaboration technology (Calefato et al., 2012; Derntl, Renzel, Nicolaescu, Koren, & Klamma, 2015)

Research suggests that collaboration technologies provide help in alleviating some of the distributed collaboration challenges and support the special characteristics of GSD

environments (Portillo-Rodríguez, Vizcaíno, Piattini, & Beecham, 2012) (Tell, Babar, & Grundy, 2013) (Lanubile et al., 2010). Similarly, Prikladnicki et al., (2003) in their case study of requirements management challenges in GSD report that the use of tools to enable communication and information exchange is vitally important to manage requirements in GSD. Calefato et al. (2012) report that collaborative technologies with synchronous and asynchronous communication support show potential to address some of the elicitations and negotiations challenges in distributed requirements activities. Ebert (2011) suggests that management of requirements in a GSD is an area where tool support is helpful and certainly needed. He asserts that requirements change management in GSD is impossible without automated tools, a disciplined approach to manage change that supports change impact analysis and maintaining traceability information (Ebert, 2011). On the other hand, performing RM in GSD without adequate tool support is considered extremely difficult and causes many challenges (Sinha & Sengupta, 2006).

Challenges dues to lack of adequate tool support have been reported in various studies. For example, from their case study of RE practices in managing safety requirements of a nuclear power plant in Finland Raatikainen et al., (2011) identified challenges due to lack of dedicated tools support in RM. They considered that managing various requirements relationships, hierarchies, and traceability was very difficult to address due to the absence of requirements management tools. Some challenges were also experienced in transferring knowledge, representing requirements and communicating requirements changes because of lack of adequate tool support for requirements management. Similarly, based on 30 interviews of GSD practitioners Sinha and Sengupta (2006) report that RM was the most commonly cited challenge in 14 GSD projects. The practitioners faced difficulties in gaining a common understanding of requirements and managing changes in the absence of adequate tool support. The authors not only advocated for better collaboration support in practitioners' work environment, they also proposed a tool (EGRET) to facilitate communication and awareness for collaboratively managing requirements across distributed teams. Initial trials of the tool in the industry showed positive results. In another study, Damian et al (2003) highlighted the need for tool support for distributed groups to facilitate awareness in managing requirements. They proposed a set of features that need to be available in any technology designed to support awareness need in GSD. Their reported features have been found invaluable and been developed into most of the computer supported collaboration and awareness tools

including *Team Foundation Server (TFS),Confluence's JIRA & Grasshopper,* and IBM's *JAZZ* .

In practice providing tool support that covers all aspects of requirements management is done by using a combination of technologies. Researchers and practitioners recommend using a combination of available tools to manage the whole RM process. These technologies include *collaborative development environments*, (Wolf, Nguyen, & Damian, 2008) such as *TFS* and *Jazz, integrated and collaborative development environments* such as *Eclipse* and *Microsoft Visual Studio* (Booch & Brown, 2003; Herbsleb, 2007; Tell & Babar, 2011), *collaborative technologies* such as JIRA and Wikis (Ebert, 2011; Lanubile et al., 2013; Rodríguez et al., 2010) *configuration management* support in combination with other (synchronous and asynchronous) communication technologies (Bird, Nagappan, Devanbu, Gall, & Murphy, 2009; Herbsleb & Mockus, 2003).

JIRA for example, is mainly a bug reporting and issue tracking software (Prause, Scholten, Zimmermann, Reiners, & Eisenhauer, 2008). However, a combination of JIRA and *Confluence* as a collaboration tool has been reported to adequately support requirements management activities by providing requirements related information sharing. This combination provides almost all the desired features proposed by Damian et al. (2003) such as providing requirements and people related information, assigning and managing requirements and change task responsibilities, managing and distributing impact analysis and estimation and being used as the centralized requirements repository (Prause et al., 2008).

2.3.2 Collaborative Technology Limitations

Advancement in development environments, collaborative technologies and RM tools continue to facilitate RE in GSD (Carrillo, Nicolás, Fernández, Toval, Ebert, & Vizcaíno, 2015; Portillo-Rodríguez et al., 2012). These technologies however, have not eliminated the problems associated with requirement management activities in distributed settings (Calefato et al., 2012). Based on their empirical study regarding the effectiveness of synchronous computer-mediated communication in requirements and negotiations Calefato and colleagues (2012), report that despite the advances in collaboration technologies globally distributed teams continue to face significant challenges in requirements elicitation and negotiations; essential activities for dealing with requirements change. Boehm et al point to the fact that groupware-supported methodologies are among "the hardest to get right" but an even bigger challenge is to

create the collaborative technology that supports RE activities performed by geographically distributed stakeholders (Boehm, Grunbacher, & Briggs, 2001).

Review of literature on CT use in GSD also points out several other limitations of technologies such as their coverage of RM processes, integration, implementation or deployment and their selection. For example, Portillo-Rodríguez et al., (2010) discuss two important deficiencies of existing collaborative tools and technologies 1) their coverage of software development lifecycle activities and 2) their integration with each other. They conclude that none of the studied tools apart from JAZZ, provide coverage for all phases of software development. Furthermore, tool integration is a problem because integrating collaborative tools with other tools is not easy because they are not designed with the next phase of the software lifecycle in mind.

The authors further suggest that in order to be efficient in GSD individual tools need to include both synchronous and asynchronous features. Other research studies Calefato et al. (2012) and also Calefato (2011) highlight implementation and integration issues of existing CTs. Raatikainen (2011) support this view and suggest that introducing new technology is not enough as tools alone do not solve any problem. Work environments have several systems in place and CTs need to integrate with other in place systems, support the existing processes and individuals. They consider that placing a new system in such an environment to aid requirements management requirements is not a challenge per se. It is the integration of this new tool with the current work practices, lifecycle processes and other systems which is comparatively more challenging. Calefato et al., (2012) consider that while collaborative technologies are introduced to achieve effective communication and gain the necessary support for requirements and change related activities, their deployment is not a trivial task. They further note that deciding on which communication technologies to deploy for achieving effective communication in distributed RE is not an easy task.

Kuhrmann and Kalus (2008), observed a gap "between technically mature development environments and development process support". Commercial tools such as IBM Jazz and Microsoft Team Foundation Server (TFS) are aimed at mitigating some of the issues of distributed software development by facilitating team collaboration, allowing work synchronization and offering a centralized virtual workspace. The authors argue that the design of such tools however regards development and management as separate or even independent disciplines. Noting the limitations of existing CT Kuhrmann and Kalus (2008), point out that other generic collaboration platforms (for example Microsoft

SharePoint) do not provide any process that could be followed. Furthermore, they argue that tools like IBM Jazz tend to impose a pre-defined and fairly static process which has to be followed by the organization using such tools.

Surveying awareness support of the leading application life management platforms (ALMs). Lanubile et al. (2013), noted that only one of the seven platforms provides support for all four types of awareness and even that requires improvement in social awareness support. Similarly, in their previous work Lanubile et al. (2013), also acknowledge that none of the existing collaboration development environments or tools support all the activities necessary for GSD. They argue that users are therefore required to prioritize their collaboration needs as well as tools to get support for their needs. They suggest that collaboration technology should be introduced in a stepwise fashion, where you start off by using a collaboration platform and then gradually move on to sharing applications later (Lanubile et al., 2010). Tell and Babar (2011), consider that existing tools are based on the most commonly used desktop metaphor and therefore have several limitations in terms of their support for communication, collaboration, coordination and awareness among distributed team members. Tell and Babar (2011), propose that an Activity-Based Computing (ABC) paradigm can be leveraged to address many of the limitations found in current GSD tools. They claim that an infrastructure support based on ABC principles for the existing GSD tools can potentially address the tool-related challenges of GSD (Tell & Babar, 2011).

Is RM being adequately supported by existing tools and technologies? To gain insights about the degree of tool support provided for RE (including RM) processes a recent survey of RE tool vendors was conducted by (de Gea et al., 2012). Their results (of 38 tool vendors) suggest that requirements elicitation is the best supported category whereas modelling and management of requirements are the “most badly supported categories” by the existing tools, exhibiting a certain margin for improvement to support distributed requirements management (de Gea et al., 2012). Similarly, Johansson and de Carvalho (2010) based on their study of requirements management tools in the area of Enterprise Resource Planning (ERP) conclude that the existing RE tools do not support requirements management in the (ERP) systems context adequately. Another study by Vibha and Bikram and Satish (2006), discuss the limitations of existing technologies. They explain that in today’s GSD projects stakeholders cannot collaborate effectively using existing tools (Vibha, Bikram, & Satish, 2006). The available tools, although good for basic collaboration, lack adequate integration between requirements and the communication

environment which leads to information fragmentation and context switching across several media. This results in a reduced common understanding of requirements (Vibha et al., 2006). They argue that because of inadequate social contact and lack of adequate tool support geographically distributed practitioners face trouble gaining a consistent understanding of requirements or managing requirement changes.

Sinha et al., (2006), envision future tool support for distributed requirements management collaboration that includes both formal (structured) and informal collaboration services, awareness support features and knowledge management techniques built into the collaborative development environments. They argue that research initiatives, both commercial and open source have resulted in bringing elements of collaboration in the development activities, the tools however are still focused on distributed bug tracking, coding and project management. They recommend that more attention to be placed on distributed requirements related process activities. For requirements management (Damian et al., 2003) also note the limitations of existing generic awareness tools and suggest that such tools do not provide information relevant to the requirements management process in globally distributed environments. They contend that successful distributed requirements management is only possible when the tools can provide adequate information regarding the artefact involved in RM.

So the question arises, do we need more tools or better technology? The answer is not simple. Smite (2014) contends that although tools are helpful, implementing better or more tools is not going to fix most of the challenges faced in distributed development because the nature of these challenges is related to managing soft factors (Ruhe & Wohlin, 2014). Similarly, Damian (2007), suggests innovative processes and technologies are needed to manage stakeholders' expectations and interaction in global projects because it is the 'soft side'; the people in such collaboration which have to be better managed.

In contrast to the earlier optimistic views that the use of modern information technology and communication systems will render geographical distance irrelevant to business (Cairncross, 1997), researchers more recently are realizing the limits of technology effectiveness. Researchers concede that the two main challenges in GSD are insurmountable using technologies: time-zone and cultural differences. For example, Prikladnicki, Marczak, Carmel, & Ebert (2012), consider that there is no miracle technology that can overcome time-zone differences. They argue that even the fanciest video conferencing technology or 3G holography cannot overcome the basic time-zone problem- which is the "sleep time on the other side of the world". Similarly, Lanubile et

al. (2013), contend that technology cannot overcome cultural challenges and poor management however being aware of what is going on in the group and making good use of technology can help mitigate challenges of distributed projects.

Thus, the opinion on whether CT is adequate to manage requirements is divided among the researchers. To explore this further, the next section (2.3.3) suggests that existing tools are adequate to handle challenges of GSD whereas section (2.3.4) deals with those studies that suggest that the use and experience of existing tools and technologies is not always positive.

2.3.3 Adequacy of Existing CTs

There are several studies which claim that the collaborative technologies generally available for GSD project are sufficient and there is no need for new CT technologies to be developed. Here, some of these studies are reported to support such claims. Bjorn et al., (2014 a), provide a comparative analysis of four ethnographic studies in GSD and reports on the use of collaborative technologies and their impact on distributed collaboration. The aim of this study was to identify challenges people face while working in GSD which could then inform the design of new collaborative technologies. They assessed the role of technologies in supporting four distance concepts introduced by Olson & Olson (2000), namely *common grounds, coupling of work, collaboration readiness & collaboration technology readiness.* Their results suggest that collaborative technologies have already been made a part of people's work and individuals are comfortable (and often quite familiar) with these technologies. The participants did experience some challenges with a) instability of CTs (such as video conferencing) which required much effort to establish cross site connections and b) some issues related to organizational practices (such as entering video conferencing rooms without access, remote control or booking that resulted in postponing some meetings). However, the research participants clearly suggested that there were no problems that required new technologies as a possible solution. The authors thus conclude that in (globally) distributed settings collaboration technology readiness (CTR) takes a different shape. We therefore, need to refine our arguments concerning CTR and re-evaluate this factor when thinking about future CSCW systems design.

Derntl and colleagues (2015) outline common challenges of distributed collaborative research projects and based on the open source development model propose a methodology along with some tool related recommendations to overcome collaboration challenges. Their recommendation in the area of RM tools is motivated by the survey by

de Gea, Nicolas, Aleman, Toval, Ebert, and Vizcaino (2011). Their survey suggests that there is a margin for improvements in the existing tools of RM. Derntl et al. (2015) recommend JIRA to overcome challenges with the management of issues, including bugs and change requests. Although they do not specify exactly how, in their experience they found JIRA to be better than other issue tracking software. They do mention the application of JIRA by major Open-Source Software (OSS) organizations such as the Apache Foundation to convey its usefulness and importance (Derntl et al., 2015). It is important to note that instead of designing new tools they consider that using existing tools and technologies for code, issues, requirements and quality management can help overcome distributed collaboration challenges.

Findings from the case study by Wolf, Nguyen, and Damian (2008), about the effects of distributed collaboration in the *JAZZ* project suggest that geographical distance does not introduce significant delays in communication and task completion. In their study, *JAZZ* (an Eclipse based) collaborative environment mitigated some of the distance effects by facilitating distributed teams with the ability to asynchronously comment and track activities of work items. The authors suggest that the utilization of a shared repository and the ability of asynchronous commenting enhanced knowledge exchange among team members. These mechanisms along with processes and practices facilitated effective cross-site collaboration and did not allow distance to have a significant impact on collaboration activities.

Tell and Babar (2012), observed that commonly available and simple standalone tools were the most favoured by the GSD teams instead of heavyweight tools specifically designed for collaboration and requirements management. They observed tools such as Microsoft Excel were commonly used for managing requirements as well as bug or issue tracking (Tell & Babar, 2012). The practitioners were keen to use standalone tools even when they did not provide any kind of computer mediated team work. Ebert (2012) also reports that they have observed standalone tools such as spreadsheets to manage all requirements in certain agile projects. Herbsleb (2007), similarly reported that ordinary tools render themselves quite well to support requirements related activities up to a certain point in even in GSD environments. Baldwin and Damian (2013) in their experiment with three GSD teams in a student project demonstrated that lightweight communication tools such as chat and email outweighed specifically designed collaboration tools such as IBM's Jazz. These studies hint towards the adequacy of existing CTs and tools.

2.3.4 CTs Experiences in GSD- Positive and Negative

Despite the claims of their adequacy and supportive role of the available CTs, the application of different communication and collaboration media in GSD often results in mixed experiences as noted in these studies (Bird et al., 2009; Herbsleb & Mockus, 2003; Wolf et al., 2008; Zowghi, 2002). A recent example of challenges faced with the use of collaborative technologies is shared by (Esbensen, Tell, Cholewa, Pedersen, & Bardram, 2015). In their studied distributed agile development project they noted a number of recurring challenges with existing collaborative technologies. They observed that handling tasks on the physical scrum board and setting up daily stand up meetings were found cumbersome due to lack of proper tool support at the right location. The challenges reported were, having to go to a dedicated videoconferencing room at each site, setting up a conference call and the needed software development tools. Due to the use of the HP Application Lifecycle Manager (ALM) not being designed for collaborative use, the amount of work needed to set up a screen alongside sharing participant videos (through VPN and Remote Desktop Sharing) was extremely demanding. When the experienced developer was not around other team members struggled to get the video conference up and running. According to the authors the meetings did not last to their desired duration and were far from ideal.

In the same study lack of awareness was reported as another recurring challenge. It was related to bug fixing (sometimes also considered a requirements change (Somerville 2012)) where two developers were working on the same bug. According to the authors, the unavailability of a digital scrum board resulted in lack of a bug's status change awareness. This lack of awareness had to be accounted for by using instant messaging on a daily basis to update the change status to the lead developer on the remote site. However, as a consequence the problem was created where this information only resided with the developer and mutual awareness for the other team members could not be maintained. Motivated by these challenges they designed a digital board (dBoard) to facilitate collaboration in such projects. Their initial user study results show that the dBoard was found useful and easy to use by the participants.

In another recent study, Razzak (2015) reported mixed experiences from their research participants about knowledge management practices and the use of supporting tools in seven globally distributed agile projects. Different teams in these projects applied different practices and tools to manage project knowledge. It was interesting to note that two teams with almost identical knowledge sharing tool sets reported totally different

experiences of their CT use. While one of the managers was quite satisfied with the tool set the other was quoted as saying *"I am not satisfied with the current tools. It is tough to describe designs to new team members"* (Razzak ,2015).

Another example of the challenges faced with collaborative technologies is shared by Kääriäinen and Välimäki (2009). They report from their industrial study of Application Lifecycle Management (ALM) improvement experience in their case company that was aimed at providing better support for the company's global development teams. They report the experience of two teams piloting the ALM who ended up having to apply two very distinct solutions due to the deficiencies of ALM platforms. They conclude that one ALM solution does not necessarily suit all teams in an organization. They suggest that the lack of integration among the available commercial tools (such as Team Foundation Server (TFS) and Microsoft Notes Database in the company) did not allow the company to manage all lifecycle artefacts, a problem specifically faced by Team 1. For team 2, the organizational template that came with the ALM was not specific to the project characteristics and needs. In order to avoid building a template from scratch or making modifications to the template that would have required significant effort, the team had to use the basic template that came with ALM which was suboptimal. They conclude that integrating different commercial tools is difficult so it is better to stick with one technology (for example TFS, Microsoft Notes and IBM's Lotus). They do note that the ALM was found to be an important technology in supporting GSD by providing visibility and consistency of project information.

The successful use of CT's in GSD has also been reported. For example, Sinha et al 2006 report a successful use of their proposed tool (EGRET) to support requirements management and collaboration in GSD projects. Similarly, Moore, Reff, Graham, and Hackerson (2007) have previously reported a successful use of TFS to support Scrum methodology, however that too required a considerable tailoring of the TFS process template to make it fit for their purposes.

Having discussed the two differing opinions regarding the usefulness and adequacy of collaborative technologies this research attempts to understand the actual role played by CTs in managing requirements change. The specific research question relating to the role of CTs in managing lifecycle activities of a change is reported at the start of Chapter 3.

In light of the above discussion, the next section establishes the need to explore RCM with a lifecycle perspective of a change in combination with CTs.

2.4 Why this Study

Global software development today is an "established, emergent and growing enterprise" (Bjorn et al., 2014) and so is its failure rate. More than half of all distributed projects fail wasting substantial human effort and resources (Lanubile et al., 2013). One of the primary reasons of this failure is poor management of requirements and their evolution (Ebert, 2014; Lai & Ali, 2013). Significant research effort is being invested which has resulted in many proposed solutions but the challenges have not been overcome (Schneider et al., 2013). Researchers have been trying to tackle these problems from various process models (Hussain & Clear 2012; Lings et al., 2007; Niazi et al., 2008), tools (Pirkkalainen & Pawlowski, 2013; Portillo-Rodríguez et al., 2012), practice oriented solutions. However, in reality, we still lack the understanding about how geographically distributed teams actually manage requirements evolution and the information about change (Damian, Marczak, Kwan, & Ebert, 2012, p. 258). The sheer volume of failure and our lack of understanding in the area of requirements change management thus motivates the study.

In GSD projects, changes in requirements are the predominant reason for distributed collaboration that require tool support to manage change related information (Damian et al., 2012, p. 261). Rapid advances in communication and collaboration technologies have made GSD a reality (Bass 2015), but the challenges and failures persist. There is a lack of understanding of appropriate technological capabilities and usefulness of CT application to overcome RM challenges in GSD. Researchers' opinion is divided and there is a relative paucity of evidence from practice about the usefulness of CT in resolving the issues in GSD especially for RM (Baldwin & Damian, 2013). The pervasive use CT and the mixed results (positive and negative) on their effectiveness in practice demand further investigation to gain greater understanding of the role and impact of collaborative technologies in problem solving and social interactions within teams as well as across organizations (Andres, 2013). Mistrik et al. (2010), consider collaborative software engineering is a very heavily researched yet challenging area in terms of adoption of collaboration practices, processes and tools as well as improving state-of-the-art. They conclude, "we still do not know the ideal way to share knowledge, facilitate the most effective communication, co-ordinate massively distributed work, and design and deploy support tools for these activities" (Mistrík, Grundy, van der Hoek, & Whitehead, 2010). Better understanding and proper use of the existing and available technologies may help in addressing some of the GSD challenges (Prikladnicki et al., 2012). This again

motivates this study to explore and understand the actual role played (helpful or hindering) by collaborative technologies to ascertain if more technological solutions are required to solve RCM challenges in GSD.

Verner and Abudllah (2012) as well as Hiisilä, Kauppinen, and Kujala (2015) emphasise understanding RE challenges in outsourcing from a client perspective and argue that no straight forward RE text book solutions are available to answer these issues. According to them, to address the challenges and develop new processes and practices there is a compelling need to understand customer organizations' RE processes and practices in outsourced environments (Hiisilä et al., 2015). This thesis on the other hand posits that both the client and the vendor perspectives need to be assessed. A comparison of both these perspectives can actually provide better insights into the realities of the challenges- as sought in this thesis. Although researchers have been investigating the area of requirements management in GSD for over a decade, there is relative paucity of research into managing lifecycle activities of a requirements change briefly covered in for example (Carlshamre & Regnell, 2000; Grehag, 2001).

Similarly, although tool support for managing requirements and change in GSD has received substantial attention (Kelanti M et al., 2013; Rodríguez et al., 2010; P. Tell & Babar, 2012b) however their role in RE lifecycle activities has not been adequately explored. The two notable exceptions are (Carlshamre & Regnell, 2000) and (Grehag, 2001). The study by Carlshamre and Regnell (2000), does talk about the lifecycle approach for requirements management however is different to this research in at least two accounts 1) they have studied the industrial market-driven requirements engineering processes which is related to requirements rather than changes in requirements 2) it does not explore the role of collaborative technologies in managing requirements especially changes. Similarly, the study by Grehag (2001) considers that requirements management is challenging and identifies five areas needing further research. She proposes a lifecycle approach of managing requirements throughout the whole lifecycle of the information system whereby a set of initially agreed *'REQUIREMENTS'* (See Figure 2.2) has to be maintained throughout the system's lifecycle. Grehag, however does not specify the actual lifecycle activities of a requirements change and its treatment, neither does it talk about the role of collaborative technology in change management activities.

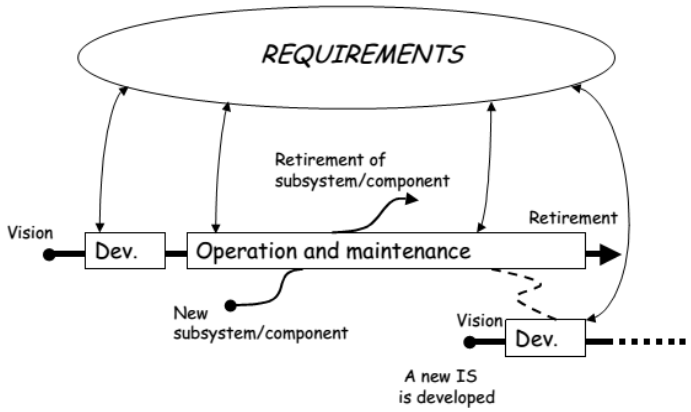

Figure 2. 2 Requirements Management from a life cycle perspective (Grehag, 2001)

This thesis thus posits that RCM has not been studied using a lifecycle perspective approach in combination with the role of CTs in each lifecycle phase. This research tries to bridge that gap by looking specifically into the challenges of managing requirements change by following its lifecycle in a GSD environment and the associated role of collaborative technologies in managing change. Many partial solutions have been proposed to address the challenges posed by carrying out RE in GSD however none of them are complete enough to cover the complete lifecycle of RE (Lopez et al., 2009). Similarly, the practices, processes and tools for traditional product and software development do not support globally distributed product development (Parviainen & Tihinen, 2011).

Despite the works just described and the plethora of tools available (Birk & Heller, 2015) there remains a shortage of meaningful empirical research on how organizations perceive and support the critical process of requirements change management in GSD projects. Rather than immediately targeting tool development, this research work takes a different approach- it adapts a lifecycle perspective, to understand the challenges of managing requirements change in GSD environments and explore the intertwined relationship of process, practice and the 'assumed' role of collaborative technology in managing changes. This study posits that such an approach is required to address the issues faced in requirements change management because currently they are not well understood, there is a lack of supporting empirical evidence for the proposed solutions and there are no theories available to resolve them.

2.5 Chapter Summary

This chapter has reviewed the existing literature in the area of requirements change management in global software development. It has highlighted the persistent challenges

that need to be resolved in this area and the need to further understand and explore how requirements change management is performed in GSD. The chapter has reviewed the literature on the available collaborative technologies and tools and the differing views on the adequacy and usefulness of these tools in supporting requirements change management. The chapter concludes with the identification of an area within requirements change management that is relatively unexplored and under researched. It also identifies the novel approach (the lifecycle activities of a requirements change and the associated role of collaborative technologies) taken to explore requirements change management in GSD.

The following chapter (Chapter 3) describes the research methodology and design along with the approaches to analyse the data.

Chapter 3. Research Design and Methodology

In this chapter, first the research study design is presented. Then the rationale for selecting a qualitative research methodology and the context of the research is provided. Next, the selected cases, research participants and data collection methods are discussed in detail. Furthermore, details about the analysis techniques applied and the tool used in the research. Finally, the chapter discusses how the research trustworthiness was ensured, its limitations and ethical considerations. The fundamental goal of describing all these activities is to ensure that research objectives, as laid out in Chapter 1, are seen to be satisfied in a robust and rigorous manner, and also to address the resultant research questions in a similarly justified and defendable manner.

The questions this research tries to answer are presented here to facilitate the understanding of their relevance to the adopted research design and applied research methods.

- Research Question 1: *What are the challenges faced by practitioners involved in managing lifecycle activities of a requirements change in a globally distributed software development?*
- Research Question 2: *What is the helpful or hindering role of collaborative technologies in managing lifecycle activities of a requirements change?*
-

Based on the nature of these research questions, this research follows a qualitative design and adopts interpretivism as a worldview to understand and explore the perceived role of collaborative technology and the challenges faced by individual team members in global software development (GSD) while managing requirements change activities.

The inductive and qualitative stance of research is aimed at better understanding human thoughts and actions in a social and organizational context and to produce deeper insights in the studied phenomenon (Klein & Myers, 1999). The chosen interpretive qualitative approach thus appropriately defines how data was collected, analysed and interpreted. The next section 3.1 introduces research paradigms and their influence on research design.

3.1 Qualitative Research Design

This research work employs qualitative research design to explore, understand and explain a social phenomenon (Klein & Myers, 1999). The research design presented here delineates the plans and procedures that cover decisions from "broad assumptions to detailed methods of data collection and analysis" (Creswell, 2009, p. 87). The main factors considered for the selection of this research design were the research problem,

degree of researcher control over behavioural events, and focus on the contemporary phenomenon (Creswell, 2009, p. 87).

As a blueprint of research, a qualitative design guided me to answer questions such as what questions to study, what data were relevant, what data to collect and how to analyse the results (Philliber, Schwab, & Samsloss, 1980). For this study both qualitative and empirical data such as interviews, requirements change management process data and artefacts were collected from the selected projects and utilized for further analysis. Analysis of data was carried out in two stages; first by mapping the existing RCM process to understand the phenomenon and secondly applying the same process mapping to perform analysis on the lifecycle activities of managing requirements change. A thematic content analysis approach (Braun & Clarke, 2006) was adopted in this study to draw conclusions based on the aggregation of the identified themes from qualitative data collected.

3.2 Research Paradigm Selection

Choice of a paradigm makes explicit the intent of, motivation for and expectations from a research study (Mackenzie & Knipe, 2006). In (scientific) research, these choices are linked to a researcher's worldview about the nature of the world (ontology), nature of knowledge (epistemology) and how knowledge can be sought (methodology) (Guba & Lincoln, 1994). Based on the combinations of these distinct ontologies and epistemologies various research paradigms exist– the traditional or ''classical'' science 'objective' paradigm, the social sciences 'interpretive' paradigm, and the critical sciences 'evaluative' paradigm (Clear & MacDonell, 2011). It is argued that a researcher's worldview or paradigm may influence the "exact nature of the definition of research" (Mertens, 2005, p. 2).

3.2.1 Interpretivist Research Paradigm in this Research

This research work is orientated towards a social science's 'interpretive' paradigm suggesting that in scientific terms what forms a truth is subjective and uncommitted neutrality is not possible. The researcher believes that multiple realities of the same phenomenon can exist that are subjective constructions of the mind and perceptions of reality vary when viewed through the lens of different languages and cultures (Guba & Lincoln, 1994). Furthermore, richer findings emerge as a result of the interaction between researcher and the situation where the values and beliefs of the researcher act as central mediators (Fitzgerald & Howcroft, 1998; Mackenzie & Knipe, 2006; Walsham, 1995). In

agreement with the argument of Orlikowski and Baroudi (1991) that social reality can only be 'interpreted' not "discovered"; this study adopts a research strategy which is qualitative, interpretive and exploratory in nature (Creswell, 2009, p. 87).

The primary interest of this study was the understanding of human thought and actions in a social and a global software development organisational context by relying on the participants' views on the situation being studied (Klein & Myers 1999). Interpretivism was used as a typical approach to qualitative research for addressing the processes of interaction among individuals (Creswell 2009).

Both the phenomenon and its perceptions (RCM process and the perceived challenges along with the role of collaborative technologies) focused in this research had an empirical as well as 'experienced' dimensions. This qualitative interpretive research strives to understand and explore (through the participants' own interpretations) how geographically distributed team members work to make sense of their collective practice and what is the perceived role of collaborative technology in RCM process.

3.2.2 Interpretivism Vs. Other Research Approaches

In direct contrast to the beliefs in a positivist paradigm which assumes an "objective" or "factual" universal account of events and situations, this interpretive study instead seeks a relativistic (though shared) understanding of the phenomenon (Orlikowski & Baroudi, 1991). The 'received' *positivist* approach of science that aims to predict and control natural phenomena are not seen as feasible for this research because statistics and other quantitative methods may not adequately describe and explain human behaviour (Prikladnicki, Damian, & Audy, 2008). The researcher relied heavily on the understanding of organizational contexts, processes and the work practices involved in RCM. This meant that the research methods applied had to be purposefully chosen such that they are contextually dependent and observational in nature rather than being oriented towards *experimentation* (Davies & Nielsen, 1992). Individual meanings were needed to be critically examined and understood to answer the research questions about the challenges faced by people and the role of collaborative technology. Therefore, an interpretive approach and data were considered better suited as they can provide richer insights into human behaviour (Guba & Lincoln, 1994; Strauss & Corbin, 1990).

Given the collaboration intensive and globally distributed nature of the studied phenomenon (RCM process activities) in this study, the researcher shares the view of (Orlikowski & Baroudi, 1991) who believe that "the design and use of information technology in organizations, in particular, is intrinsically embedded in social contexts,

marked by time, locale, politics, and culture". Similarly, *Positivism* was not considered a suitable approach because it "disregards these historical and contextual conditions as possible triggers that influence human action. However, neglecting these influences may not reveal a complete picture of information systems phenomena" (Orlikowski & Baroudi, 1991).

The *interpretive* stance of this study is based on the interpretive epistemological belief that social process may not be adequately captured in hypothetical deductions, rather it can be better understood and explained by getting inside the world of those generating it (Rosen, 1991). In-line with the philosophical orientations of a qualitative paradigm, this research is conducted with the belief that all quantification is limited in nature, because it relies upon studying only one small portion of a reality that may not necessarily be split or unitized without losing the importance of the whole phenomenon (Krauss, 2005).

The exploratory nature of the study with the aim to understand the phenomenon runs contrary to the *critical paradigm approach* which seeks to challenge the status quo and tries to intervene and transform the reality under study. Similarly, as opposed to the objective of critical studies that strive to create awareness and understanding of the various forms of social domination in order to make it possible for the people to eliminate them, the nature of this study is to understand the reality.

Although the larger aim of the study involves providing some guidelines or recommendations that may address some of the challenges faced by the practitioners, however this study still does not align with a *critical paradigm* as it is neither about social domination, or challenging the status quo, or rectifying structural contradiction in the social system.

The interpretive perspective of this study, therefore attempts "to understand the intersubjective meanings embedded in social life [and hence] to explain why people act the way they do" (Gibbons, 1987, p. 3). Table 3.1, provides the dichotomy of major paradigms which is meant to explain and compare various dimensions of these paradigms. The research approach chosen in this work is because of its apt mapping with the characteristics noted in all four interpretive dimensions i.e. ontological, epistemological, methodological and axiological levels (shown on the left side of Table 3.1).

PARADIGM LEVEL	
Constructivist	**Positivist**
No universal truth. Understand and interpret from researcher's own frame of reference. Uncommitted	Belief that world conforms to fixed laws of causation. Complexity can be tackled by

neutrality impossible. Realism of context important.	reductionism. Emphasis on objectivity, measurement and repeatability.
ONTOLOGICAL LEVEL	
Relativist Belief that multiple realities exist as subjective constructions of the mind. Socially-transmitted terms direct how reality is perceived and this will vary across different languages and cultures.	**Realist** Belief that external world consists of pre-existing hard, tangible structures which exist independently of an individual's cognition.
EPISTEMOLOGICAL LEVEL	
Subjectivist Distinction between the researcher and research situation is collapsed. Research findings emerge from the interaction between researcher and research situation and the values and beliefs of the researcher are central mediators.	**Objectivist** Both possible and essential that the researcher remain detached from the research situation. Neutral observation of reality must take place in the absence of any contaminating values or biases on the part of the researcher.
METHODOLOGICAL LEVEL	
Qualitative Determining what things exist rather than how many there are. Thick description. Less structured and more responsive to needs and nature of research situation. **Exploratory** Concerned with discovering patterns in research data, and to explain/understand them. Lays basic descriptive foundation. May lead to generation of hypotheses. Induction Begins with specific instances which are used to arrive at overall generalisations which can be expected on the balance of probability. New evidence may cause conclusions to be revised. Criticised by many philosophers of science. But plays an important role in theory/hypothesis conception. **Field** Emphasis on realism of context in natural situation, but precision in control of variables and behaviour measurement cannot be achieved.	**Quantitative** Use of mathematical and statistical techniques to identify facts and causal relationships. Samples can be larger and more representative. Results can be generalised to larger populations within known limits of error. **Confirmatory** Concerned with hypothesis testing and theory verification. Tends to follow positivist, quantitative modes of research. **Deduction** Uses general results to ascribe properties to specific instances. An argument is valid if it is impossible for the conclusions to be false if the premises are true. Associated with theory verification /falsification and hypothesis testing. **Laboratory** Precise measurement and control of variables, but at expense of naturalness of situation, since real-world intensity and variation may not be achievable.
AXIOLOGICAL LEVEL	
Relevance External validity of actual research question and its relevance to practice is emphasized, rather than constraining the focus to that researchable by 'rigorous' methods	**Rigour** Research characterized by hypothetico-deductive testing according to the positivist paradigm, with emphasis on internal validity through tight experimental control and quantitative techniques.

Table 3. 1 Research Paradigms Comparison Constructivist versus Positivist (Fitzgerald & Howcroft, 1998)

3.3 Research Methodology

A qualitative research study can be conducted using the many available flavours of qualitative research methodologies (such as the 19 strategies of inquiry stated by Wolcott 2001 cited by Creswell 2009, p.12). Among the commonly known and reported strategies of qualitative strategies of inquiry or methodologies are Ethnography, Grounded Theory, Case Study, Phenomenological Research, and Narrative Research (Creswell, 2009). Each strategy has its own strengths and weaknesses making none of them a perfect fit for every case or universally applicable (Gill & Johnson, 1991) Cited by (Ó Conchúir, 2010).

This research explores the phenomenon of interest in two cases in real life settings to seek new insights and generate ideas for new research (Runeson & Höst, 2009). An exploratory multiple case study methodology therefore is selected for this research to compare and contrast two different cases. The requirements change management process is selected as the common unit of analysis in both cases. The methods for data collection include semi-structured interviews, electronic and non-electronic artefacts from RCM process and observations. A thematic content analysis is chosen for the analysis of collected data. Figure 3.1, shows the methodological design framework which is based on Creswell (2009) which is adapted to show various elements of the research design used in this study.

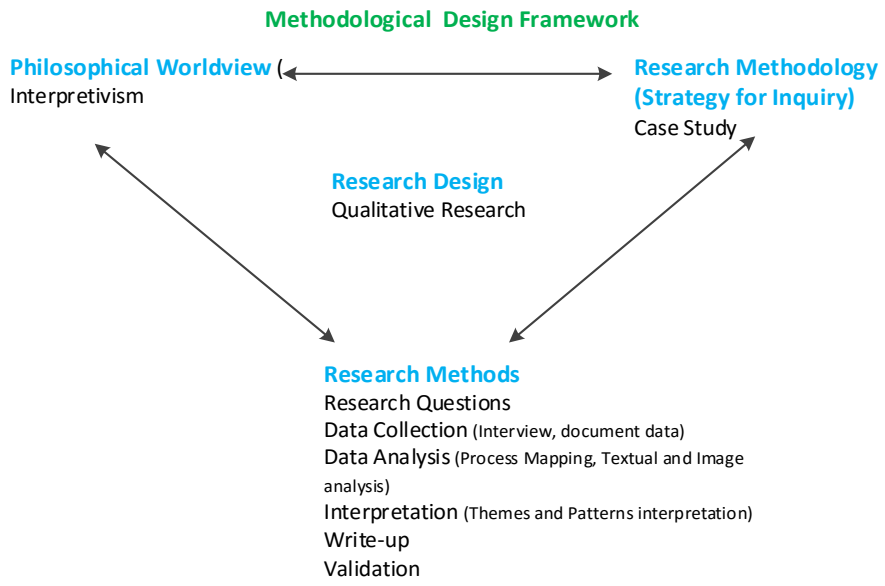

Figure 3. 1 Research Design Framework (Adapted from Creswell 2009)

3.3.1 Selection of Exploratory Case Study Method

The phenomenon under scrutiny in this research involves studying human behaviour, their interactions, perceptions, languages and culture in real life settings. Therefore, an interpretive approach is deemed appropriate as advocated by the scholarship in qualitative research (Darke, Shanks, & Broadbent, 1998; Easterbrook, Singer, Storey, & Damian, 2008; Klein & Myers, 1999).

The limited coverage of managing lifecycle activities of a requirements change in GSD e.g. (Carlshamre & Regnell, 2000), merits an exploratory approach to better explore and understand the phenomenon of interest (Runeson & Höst, 2009; Yin, 2003). An in-depth exploration of the practices employed by GSD practitioners requires a valuable 'thick' description essential to understanding of the context and situation. Such an outcome is generally expected from an exploratory case study (Lincoln & Guba, 1990). A case study methodology is thus seen as an enabler for the researcher to gain deeper understanding of the phenomenon of interest (Runeson & Höst, 2009; Yin, 2014).

The choice of a qualitative interpretive exploratory case study for this research was influenced by these factors: goodness of fit with the research objectives, past experience of using this method, time constraints, the researcher's philosophical beliefs, the nature of the phenomenon being studied and access to suitable data (Yin, 2003). Moreover, the (exploratory) nature of research questions posed in this research, limited control of the researcher over actual events and the focus on a contemporary event further influenced my methodology selection Yin (2014, p. 9).

This selection is in line with Yin's arguments that the exploratory nature of a questions, such as *'what',* espouses an exploratory study. He further argues that such questions are a "justifiable rationale for conducting an exploratory research" (Yin 2014, p.10). On the other hand, a quantitative approach, would have been more appropriate if the research questions were of the form of "how many" or "how much" (Yin 2014, p. 10).

Given the limited time and resources, an ethnographic approach was not deemed feasible. The exploratory case study methodology on the other hand would enable the researcher to elicit qualitative data from participants while not participating directly in the phenomenon over a prolonged period of time, (Creswell, 2013). A case study approach is also chosen because of the flexibility it offers to the researcher to acknowledge multiple realities, multiple meanings and findings that are observer dependent (Yin 2014, p 17). It led me to ask the research questions such as (what, when and how) which could be answered by gradually making sense of a social phenomenon by comparing, contrasting,

replicating, cataloguing and classifying the object of study (Flyvbjerg, 2006; Miles & Huberman, 1984; Yin, 2014). The strength of case study method, which makes it a suitable choice for this research, lies in its versatility that it can accommodate both a realist as well as a relativist perspective (Yin 2014, p 17) that can account for multiple realities as desired in this study.

Table 3.2 lists a variety of possible research methods depending upon the factors discussed above.

Method	(1) Form of Research Question	(2) Requires Control of Behavioural Events?	(3) Focus on Contemporary Events?
Experiment	How, why?	Yes	Yes
Survey	Who, what, where, how many, how much?	No	Yes
Archival Analysis	Who, what, where, how many, how much?	No	No/ Yes
History	How, why?	No	No
Case Study	How, why?	No	Yes

Table 3. 2 Relevant Situations for Different Research Methods (Yin 2014, p.9)

Use of exploratory case study is repeatedly employed in GSD research (Barney, Wohlin, Chatzipetrou, & Angelis, 2011; Hossain, Babar, & Paik, 2009; Hussain & Clear, 2012; Prikladnicki, Audy, Damian, & de Oliveira, 2007; Prikladnicki et al., 2008; Richardson, Avram, Deshpande, & Casey, 2008; Sa, Marczak, Antunes, & Audy, 2003; von Stetten, Beimborn, Weitzel, & Reiss, 2011; Zahedi & Babar, 2014). More specifically, distributed requirements engineering and management have also been investigated through case study methodologies to understand and explore social phenomena in real life context (Bhat et al., 2006; Damian, 2007; Damian & Chisan, 2006; Damian & Zowghi, 2003; Hussain & Clear, 2012b).

Two notable empirical studies in the GSD context performed in the recent past that involved requirements change data are discussed here. In the case study conducted by (Herbsleb & Mockus, 2003), the authors utilized both survey data and change request data from a change management system to prove that distance introduces delay in GSD communication and coordination work. Although the study involved Change Request data, it did not focus on the actual lifecycle activities of a change request neither did it focus on challenges other than delay. Furthermore, the role of collaborative technology

was also not analysed in that study, which distinguishes this research from the study by (Herbsleb & Mockus, 2003).

A more relevant case study of the use of collaborative technology for requirements management was carried out by (Sinha et al., 2006). They employed semi structured interview techniques to carry out interviews of 30 practitioners. Their study identified some significant challenges related to requirements management in a globally distributed environment. The main challenge identified by this case study was the communication and management of requirements. To this end they proposed EGRET, a tool to facilitate requirements management in distributed contexts. Again the study was focused on tools that provide collaborative support to the core development team and not focused specifically on the lifecycle activities of a requirements change. Also the data collected was from practitioners who were mainly from the vendor side or members belonging to development teams. They did not include client-side stakeholders and end-users who also face significant changes when dealing with the requirements change process.

3.3.2 Alternate Methods

Research methods serve complementary functions where one might cover a particular topic better than the other (Green, Camilli, & Elmore, 2012). For example, in this research a *survey* approach does not fit the context as the research aim is neither the quantification of results nor the discovery of a 'cause and effect' relationship based on generalizable statistical data (Gable, 1994). This research poses 'what' questions that demand 'thick' descriptions and rich outcomes in contrast to 'how or how many' kind of questions which a survey result may answer (Green et al., 2012). Furthermore, *surveys* may provide only a "snap-shot" of the situation at a certain point in time which may not be a rich data set as desired in exploratory research for analysis and elaboration. A thick description of the phenomenon of interest would therefore not be possible through a survey.

Since the participants' perceptions and underlying meanings were considered important a survey approach that could strip off the context to 'buy' objectivity at the cost of a deeper understanding was not desirable (Kaplan & Duchon, 1988, p. 572). Similarly, field and laboratory experiments were not considered appropriate as they are more suitable for inferring causal relationships. In order to control variables, laboratory experiments remove the phenomenon from its context; a requirement of this study. While generating hypotheses, testing or setting goal statements might sharpen the focus of the study, experiments at the same time, minimize the interest in the situation and circumstances (Stake, 1995) and hence were not considered suitable for this study. In contrast a

qualitative exploratory design was desired since it can "appreciate the uniqueness and complexity of the case, its 'embeddedness' and interaction with its context" (Stake, 1995).

The nature of this exploratory study and the epistemological stance (interpretivism) requires flexibility and lesser control over behavioural events. Therefore, *experiments* were ruled out since they were incompatible with the epistemological standpoint at the core of this research (Creswell, 2013).

Archival analysis was also not useful because it would have failed to include the circumstantial and contextual input provided by the participants. Relying on "derived" data such as test results and other available statistical data maintained by organizations and in responses to questionnaires is not as valuable as gaining first hand understanding of the phenomenon of interest observed in real life and natural settings (Bromley, 1986, p. 23) cited by (Yin, 2006).

Ethnography was ruled out because of the limited time and resources available to the researcher. An ethnographic approach would have required spending extended periods of time with the participating organizations which at the selected sites which was not feasible. Some research participants were mobile between sites which meant prolonged access to their expertise and time was also not possible. Similarly, high degree of access to the development teams over an extended period time was not possible hence an ethnographic approach was not deemed appropriate for this study.

3.3.3 Single case Versus Multiple Case Study Design

Multiple case studies are variants of the case study methodological framework that have recently gained more popularity and are being used more frequently (Yin 2014, p.56). The reason to choose multiple case study design for this research is that it strengthens the ability of the case study design to generalize while preserving the in-depth description (Herriott & Firestone, 1983). Furthermore, the evidence from multiple cases is considered more compelling and the overall study is considered as more robust as they "permit cross-site comparison without necessarily sacrificing within-site understanding" (Herriott & Firestone, 1983). Multiple case studies address the same research question in a number of settings using similar data collection and analysis procedures in each setting. This research acknowledges the challenges and disadvantages of the chosen multi-case study design that it is a more time consuming and expensive endeavour.

Figure 3.2 shows various possible case study designs (both single and multiple-case). In the Figure 3.2, single case study designs are crossed out as being not applicable since this research follows a multiple-case study design. Among the other available multiple case study designs *Types (3 and 4)* this study falls under *Type 4 which is circled*. As only two cases (out of the four shown) with the same context and embedded units of analysis were studied hence the other two boxes depicted in *'Type 4'* design are crossed out. Globally distributed (Information System) Development is seen as the *Context* in this research with the *Main Unit of Analysis* being the process of requirements change management and the *Embedded Unit* being the Use and Role of Collaborative Technology (Yin, 2014).

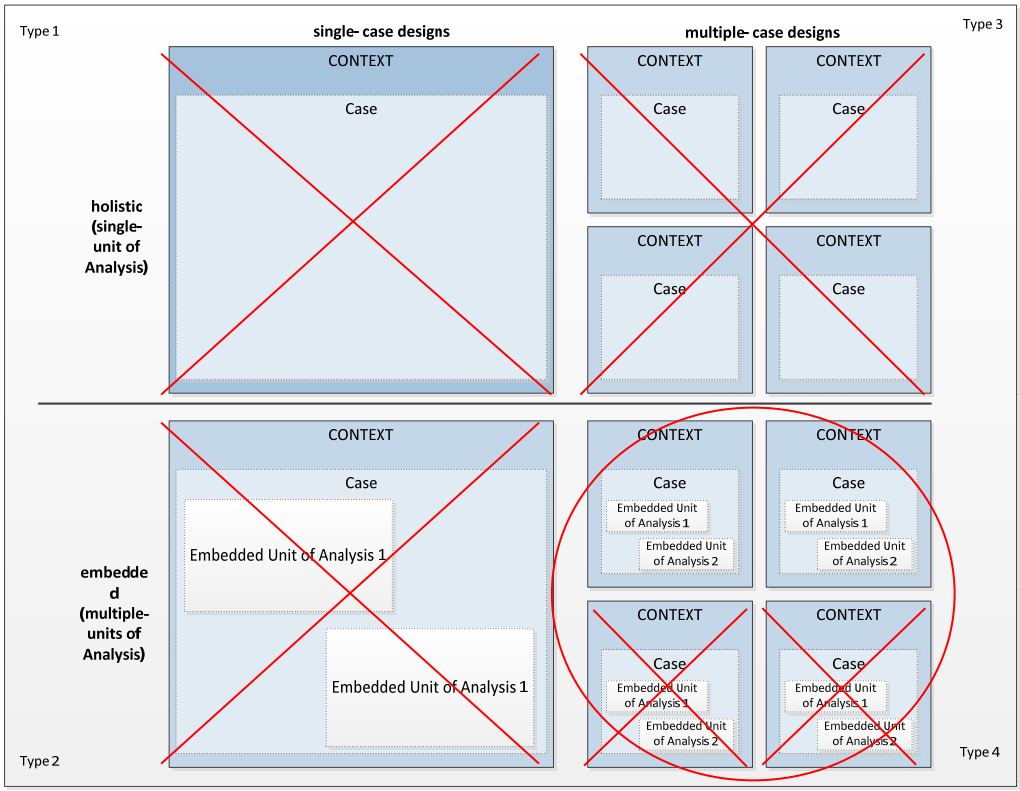

Figure 3. 2 Types of Case Study Designs (Yin, 2014)

3.4 Research Methods

Research methods used in this research, as part of the design of this study, are discussed here in detail. These methods cover, the research questions posed by this study, data collection, data analysis and interpretation, and validation techniques.

3.4.1 Research Questions

This study aims to answer the following two research questions

- Research Question 1 What are the challenges faced by practitioners involved in managing lifecycle activities of a requirements change in a globally distributed software development?

Research Question 2 What is the helpful or hindering role of collaborative technologies in managing lifecycle activities of a requirements change?

3.4.2 Data Collection

In this research qualitative methods and associated techniques are utilized for collecting data. Data collection for this research is based on the categorization proposed by Lethbridge, Sim, and Singer (2005), namely 1) direct method ; data collected directly from the participants through semi-structured interviews, meetings 2) already available organizational data (compiled or un-compiled) data, collected for other purposes such as requirements specifications, design specifications, change logs, reports, etc.) 3) indirect methods; which involve collection of raw data without actually interacting with the subjects during the data collection such as available artefacts, data in software tools, SVC data, recorded videos or audios etc.

Based on the suggestion by (Huberman & Miles, 2002, p. 9) this study combines data from the two primary sources a) semi-structured interviews and b) (electronic and non-electronic) artefacts as well as other data (e.g. observations). Utilization of several different sources of information to gather and triangulate data was aimed at limiting the effects of a single- data source interpretation and to fulfil the objectives of this study. The data collection process spanned over 10 months, i.e. from February 2013 to January 2014.

3.4.2.1 Semi Structured Interviews

Interviews are one of the most important tools of collecting qualitative data from research participants (Myers & Newman, 2007). The researcher used semi-structured interviews (having a structured and unstructured element) as a research instrument to understand the phenomenon of interest and gain further insights (Gillham, 2005, p. 72). The unstructured element facilitated a strong element of discovery, whereas the structured element allowed me to remain focused on analysis in terms of commonalities (Gillham, 2005, p. 72). The "semi standardized" or semi-standard nature of questions and prods also allowed the reordering and rewording of my questions to adjust the language level. It enabled me to respond to questions asked by the participants or to seek further clarifications as and when necessary (Berg & Lune, 2011).

The research method attempted to fit the predefined interest of the study into the unfolding topics rather than forcing interviewees to follow and fit their ideas in the logical order of the pre-defined questions (Gibson & Brown, 2009).

Although each interview was carried out in a conversational manner a certain set of questions were derived (See Appendix 4: Draft Interview Schedule) from the case study protocol (Yin 2003). The interview schedule was used with flexibility for example in ordering questions according to the 'natural flow' of conversation as well as in modifying CT related questions when the initial set of questions were not considered useful to obtain answers relevant for the research. As opportunities arose interviewees were probed for more information on particular points to further explore topics in a discursive manner. The participants were provided enough time to articulate and express their views on the general question which followed the theme of the research. This method allowed me to gain a deeper understanding of the participants' experiences and helped me in establishing a close relationship with participants. Due to this relationship the researcher managed to seek clarification of meanings, ask for additional artefacts when required.

Some topics of interest arising for the conversational interviewing were further explored which were not initially present in the interview schedule. As suggested by Walsham (1995) I tried to strike a balance between being passive and over directing. Sometimes I had to tactfully bring back the conversation to the interest of research and the related questions if the conversation were to drift into irrelevant topics without disrupting the natural flow. This was challenging in the initial interviews as my interviewing skills were not developed enough however as the interviews progressed my confidence and skill level improved significantly.

3.4.2.2 Interview Design

Interview design for this study was based on the study protocol (Appendix 4). The participants were purposefully chosen based on their suitability for this research (at least 5 years of software development and at least 1 year of GSD experience). Although practitioners with less experience also experience challenges, it was considered that the challenges might be due to their adjusting to the new environment or team, lack of experience or training and may not necessarily relate to GSD challenges in general. Hence to give the research more authenticity experienced practitioners were chosen for interviews.

The general pattern of interviews was to first ask the research participants about personal information and their role in the project, followed by the questions about challenges they face in managing requirements changes. Finally, they were asked to share their views on

the (perceived helpful or hindering) role of collaborative technologies in carrying out change related activities.

The participants were asked to share their views on how the existing collaborative technology supported or hindered the six core activities described in *Activity Based Computing (ABC)* (Bardram, 2009). Activity-based computing is an approach to computing that focuses on computational support for handling parallel, mobile, collaborative, and distributed activities (Bardram, 2009). The aim and underlying assumption of ABC is that a system with technology support that follows the core ABC principles will better help users to manage the complex set of actions, tools, materials, resources, and people involved in an activity.

To supplement the understanding of the role of collaborative technology in managing requirements change, some of the questions related to collaborative technologies were based on *Critical Incident Analysis* technique (CIT) as described by (Flanagan, 1954). Critical incident analysis is composed of a set of procedures for collecting observations of events and behaviour that have critical impact (both positive and negative) on a given set of situations (Rose & Schlichter, 2013). This technique was used to concretise thinking to specific cases and to aid participant's memory regarding use of technology in a particular but critical incident rather than have general opinions regarding technology use. One of the main advantages of this technique is to allow the participant to reflect on an incident which he/she considered 'critical' and it was more likely that they had a good recollection of the incident (Hanson & Brophy, 2012).

Based on CIT, as applied in this study, the participants were asked to think and describe a requirements change related incident. The interviewer then asked open-ended questions to understand the role played by technology in the participant-described critical incidence. The participants were guided by a structured sequence of questions as suggested by Rose and Schlichter (2013) regarding a) the context in which the critical incident took place b) cause of the critical incident c) description and outcome d) understanding of the individual(s) about the situation and e) steps taken to overcome or solve any observed problems.

However, after the first few interviews, the interview protocol had to be adjusted. Also the researcher had to move away from asking questions about a common critical incident (experienced by the whole team) to a more individual incident experienced by the project members. The researcher realized that the participants were finding it difficult to identify an incident that was considered common for every team member. The participants were

describing issues they felt were ‘critical’ to them rather than an incident considered critical by the whole team. Since the role played by each team member was different so a particular incident considered for one individual team member was not considered critical by other roles or the team as a whole. Therefore, after the two interviews the participants were asked to recall an incident they considered critical and comment on the role of collaborative technologies.

An interesting example of a critical incident explained that was not critical for everyone was a particular server going down in a night shift. This caused all sorts of problems for the support staff and the team manager. This issue however was not considered ‘critical’ by quality assurance team and some developers not relying on that server for their routine work. These members carried on with their work while the problem with that particular server was being fixed.

3.4.3 Sampling strategy

An opportunistic sampling strategy was used to select the cases for this study as it involved capitalizing on pre-existing relationships with the industry contacts. In Case 1 (CICP, New Zealand) my prior rapport and relationship with higher management allowed me to effectively discuss participant criteria and eventually recruit a reasonable number of suitable participants. in Case 2 (WIPS, Pakistan) the organization was selected using purposive sampling (Barbour, 2001) by utilizing my university and software engineering research group’s established network. The main criteria followed by the researcher to select participants was their experience in GSD (minimum five years). The selected participants in both cases were a mix of roles ranging from top executives, higher project management, team leads to software developers. The interviewees and cases were selected on the basis of expectations for the content of information they were likely to provide (Flyvbjerg, 2006).

The construction of the sample was progressively guided by the role coverage for RCM, with the goal of “maximizing variation” of opinions (Larsson, 2009, p. 31), searching for challenges, understanding practices to better understand the role of collaborative technologies in managing requirements change activities.

3.4.3.1 Interview Details

I conducted two rounds of semi structured interviews at two different sites; one in Pakistan and the other in NZ in the selected organizations for this research. The study involved a reasonably large set of interviewees for each geographically distributed project

(New Zealand, USA and Canada in Case 1, and Pakistan and USA in Case 2). Table 3.3 provides details about the number of interviews and the period when the interviewed were conducted at each.

Country	Round	Interview Type	Interviews	Between
Pakistan	1	Main	7	Feb 2013- May 2013
	2	Follow up	6	Sep 2013 - Nov 2013
	1 & 2	Informal	3	Feb 2013-Nov 2013
New Zealand	1	Main	9	May 2013 - July 2013
	2	Follow up	7	Nov 2013- Feb 2014
	1 & 2	Informal	1	May 2013 - Feb 2014
Total interviews from both sites			33	

Table 3. 3 Interview Details for the Two Case Studies

3.4.3.2 Interviews from Case 2 (Pakistan Site)

A formal invitation for participation (both for individuals and the company) was sent and approval was received before moving on to the interview stage. Research participants' eligibility criteria was evaluated prior to carrying out the interviews. Individual research participants and the company were assured of confidentiality of their participation and information (See Appendix 7: Ethics Approval). Since the participating individuals were not brought up in a culture of research, the ethics approval document and discussions regarding the research were really helpful in easing out any apprehensions regarding the research.

The researcher had a good relationship with the company based on a previous research project, however the research protocol for this study (Appendix 1, 2 & 3) was strictly followed to gain access and conduct interviews at this selected site. The participating individuals as well as the company management continued to be supportive throughout my research.

At this site seven members from the project core development team were interviewed twice over a period of nine (non-consecutive) months (See Appendix C for details). The CEO of the company (also acting as the client proxy) was collocated with the client in USA. The proxy client (normally based in USA) was also interviewed once during the first round of interviews when he was on his regular business visit to the Pakistan site. Interview duration for the second round was shorter than the first round since they were

mainly used as confirmation of views of the participant and verifying the initial analysis done by the researcher.

Out of the seven interviews conducted (and recorded) for the first interview round four were conducted in Urdu (the national language). Although the participants speaking in Urdu, did switch to English from time to time while explaining some technical information, the main language remained Urdu. When the interviewees felt more comfortable during the interviews they preferred using Punjabi (a local language) to express their views more openly. Punjabi is the mother tongue of almost 44% of Pakistan's population (Akram & Yasmeen, 2011) which was used occasionally to have a more informal dialogue in support of the formal interview discussion. The interviews conducted in Urdu were then translated into English and the transcripts were verified by my native Urdu speaking PhD colleagues.

During the interviews, certain cultural protocols had to be observed especially while interviewing female participants. Keeping the cultural sensitivities in mind a transparent meeting room (made of glass) was chosen for the interviews of female participants. Furthermore, interview timings had to be during the office hours as opposed to the male participants who could be interviewed at any time. This was important because traditionally women in Pakistani culture do not feel comfortable sitting alone with men even in an office environment and especially in the afterhours. For the same cultural reasons after hour office commitments are not deemed appropriate for females for example due to parental or societal pressure.

3.4.3.3 Interviews from Case 1 (NZ Site)

Similar to Case 2, research protocol was strictly observed for Case 1 in NZ for the purpose of interviews and data collection. Two rounds of semi-structured interviews were conducted with the participants. For the first round nine client stakeholders were interviewed including members of project management team, steering committee and other stakeholders. From the vendor site the Principal / Business Analyst (BA) was also interviewed once over Skype. See Appendix 6 for details.

The organization was initially contacted in December 2012 and the possibility of conducting this research work were discussed. At that point the researcher was asked to wait until April 2013 before starting the interviews. According to one of the committee members the project was "going through a tough stage". The organization was approached the second time in April to secure the research site for this study and an

approval was given by the management to conduct my research. A formal invitation and approval process was carried out for this case as was done for Case 2 in Pakistan, as previously discussed. For the first round interviews were relatively longer compared to the second series of interviews conducted at both sites (Pakistan and NZ). Second round interviews were mainly used for confirmation of participant views regarding challenges and the role of technology which they shared in their initial interviews. The shorter interviews were also used to capture any changes in practice for requirements management activities. At the NZ site, in the second round three participants were unavailable (the vendor in US, one steering committee member and a BA). Both the vendor and steering committee members were not available due to their busy schedules and time constraints. The client BA however could not be interviewed as he was no longer working for this organization at that time. Similarly, two participants from Site 1 in Pakistan were not interviewed because of their unavailability at the time See Appendix 5 for detail.

At NZ site all the interviews were conducted in English which were transcribed later. The interviews were conducted at the research site in either meeting rooms or offices of the participants. All interviews were conducted during normal working hours. Culture was not an issue in conducting interviews at NZ site.

3.4.4 Artefact Collection

The second source of data collection was electronic and non-electronic artefacts from the participating organizations. The electronic artefacts collected from various project and process related sources included:

a) *Contract documents* (request for proposal, reply to the request of proposal, Project planning documents, work breakdown structures, size and effort estimates work breakdown structures etc.).

b) *Email communication* between the i) development team and proxy client ii) vendor and client teams and iii) client and development/ support team members. Emails at the beginning of the contract provided information related to requirements, functionality and scope understanding/ clarifications. Later emails were a combination of various activities discussed during the project such as negotiation, information dissemination and so on. Email data and other requirements change related artefacts were instrumental in ascertaining and understanding actual requirements practiced across sites.

c) *Requirements and change management process artefact* such as (Requirements Change Logs (RCLs), change request forms, design specifications and templates, enhancement on design specifications, process workflow diagrams, RCM plan (including RCM models), and work breakdown structures (for cost/effort and size estimates)

d) *Testing and bug fixing related artefacts* (Bug reports, Testing and Quality assurance documents and plans, traceability documents)

e) *Collaborative platform data* (from Confluence and JIRA, Spreadsheets and/or other repositories for most of the artefacts mentioned in a, b, c and d above)

f) Online contract information (Request for Proposal (RFP), clarification Q/A emails between potential contractors and the govt. client) and information resources related to the product (operational manuals etc.).

3.4.4.1 Artefact Collection Case 1 (New Zealand Site)

In Case 1, at the New Zealand site the studied project had stored most of these artefacts on their own online Group Wiki (*Confluence*). The project team also used *JIRA* with the *GreenHopper* plug-in to manage the requirements and change related artefacts - User Stories, Tasks, and Issues. The team collaboration tool / group Wiki was structured based on the major project modules, where under each module the related electronic artefacts were stored. I was granted access to both JIRA and *Confluence* collaborative environments as well as all the material available on them.

3.4.4.2 Artefact Collection Case 2 (Pakistan Site)

In Case 2, most of these artefacts were stored onsite using file and directory system but some of the artefacts were stored in Visual Source Safe Software and MS Team Foundation Server. These artefacts were stored in various formats such as MS word, and pdf, spreadsheet (MS Excel), images (pdfs, Visual Studio Document) etc. The information stored on the company's in-house bug tracking & reporting system was also used.

3.5 Data Analysis

Data analysis is guided and assisted by adapting an existing methodological framework described in Figure 3.3 below. The detail regarding how data was analysed is given in Sections 3.5.1 and 3.5.2.

3.5.1 Framework Applied for Data Collection and Analysis

The analysis framework adopted from (Nurmuliani et al., 2004) as shown in Figure 3. 3 was used to analyse the data. It is a three stage process which begins by collecting requirements change management (RCM) process data, interview data and data related to technology usage. RCM Process data included artefacts such as change request forms, logs, requirements specifications, design specifications, prototype and mock up screens used in requirements engineering and implementation.

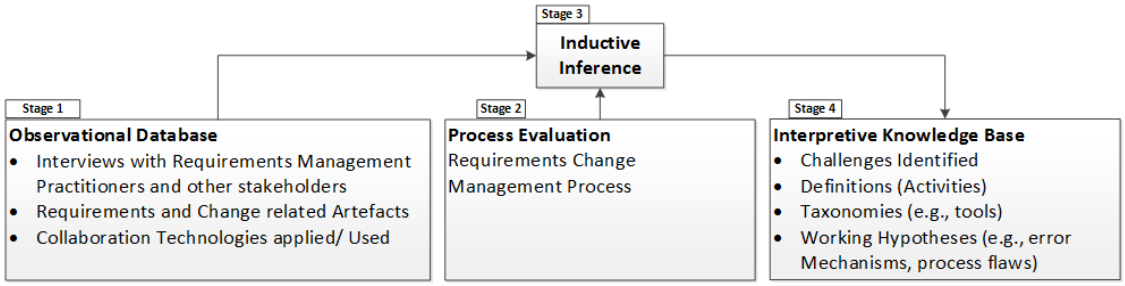

Figure 3. 3 Data Analysis Framework Adopted from (Nurmuliani et al., 2004)

The data analysis framework guided the researcher about the steps involved and their sequencing however due to the iterative nature of the data collection these steps were repeated at least twice for the two data collection cycles. Corbin and Strauss (1990) suggest that the analysis should begin soon after the first interview is conducted; thus my data analysis commenced during the fieldwork.

As evident from Figure 3.3, as a first step an observational database was populated with interviews of key participants involved in managing requirements change lifecycle activities from both the client and the vendor sites. Both project and RCM process related electronic artefacts were collected, screened, crosschecked and continuously referred to during the initial review of interview data. The purpose of such an activity is to confirm our evaluation and triangulate research findings.

The company's implemented RCM process was evaluated using available artefacts in combination with the available interview data. A special focus of this activity was to identify if any evolution in practice had occurred that may support global RCM activities. Critical incident technique (Flanagan, 1954) was used to understand helping or hindering role of the existing and available collaborative technologies and any related issues in their usage (Tell & Babar, 2011).

The participants were asked about any critical incident they could recall during the project work and were then asked to comment on the role of collaborative technology. The interviewees were further asked to comment on the capabilities of collaborative

technologies and whether they provided the collaborative support mentioned by the six core principles of Activity Based Computing (ABC) (Tell & Babar, 2011).

Data related to the technology adoption, usage and their role to support collaboration was captured and used as input for the next step in the framework (Stage 2) to perform an overall evaluation of the existing RCM processes. In an iterative manner (in stage 3) the captured data through electronic artefacts and process evaluation and mapping was then utilized to make inductive inferences about RCM process, its challenges and the role of collaborative technology. Based on the information collected in stage 2, in stage 3 the identified challenges were documented. As a final step, as shown in the framework (stage 4) these inferences collectively populated the interpretive knowledge base of identified challenges.

3.5.2 Data Analysis Techniques

The main techniques applied to analyse qualitative data collected in this study were Thematic Content Analysis (DeSantis & Ugarriza, 2000; Sparkes, 2005), Artefact analysis (Kellogg, 1990; Whitehead et al., 2010, p. xii) and (Activity Based Flow Chart) process mapping (Biazzo, 2002; Monasor, Vizcaíno, & Piattini, 2010). Contextual analysis of each studied case is carried out by adapting Activity Based Computing theory and work activity elements described in the framework proposed by Korpela et al, (2002) that are applied to understand and analyse information system development as a practice of 'real-life *work activity* in context'. The contextual analysis of each case is detailed out in Chapter 4, Section 4.2 and Chapter 5 Section 5.2.

Data analysis techniques were applied to process qualitative data that allows systematic search for *meaning* and communicate to others what is learnt (Hatch, 2002, p. 148). During the analysis the data was organized, synthesized, evaluated, interpreted and categorized in order to see patterns, identify themes and discover relationship (Hatch, 2002, p. 148) Qualitative analysis in this study was aimed at reporting findings that are in a literary style and rich in participant commentaries but grounded in the belief of multiple realities and a commitment to participant's viewpoints. Field notes and interview transcriptions were used as an aid to qualitative data analysis techniques employed in this study (Bird, 2005, p 227) cited by (Braun & Clarke, 2006).

3.5.2.1 Thematic Content analysis

In this study thematic content analysis technique is employed as a qualitative descriptive approach to analyse qualitative data (Green & Thorogood, 2013). Thematic analysis (TA)

is the most common approach in content analysis (CA). The common label of CA constitutes a number of techniques that range from the word counts of the simplest form to thematic analysis, referential and prepositional analysis (Krippendorf 1980, pp. 60-63). A thematic analysis approach utilizes the systematic characteristics of content analysis techniques but allows the researcher to combine their own meaning within a particular context. Thematic analysis is aimed at examining the commonalities, differences and the relationships among the aggregated themes generated from the collected data (Gibson & Brown, 2009).

TA is a widely used analytic method which is a flexible and useful research tool enabling a rich, detailed, yet complex account of data (Braun & Clarke, 2006). According to Braun and Clarke TA method offers, among others, the following advantages:

- Can usefully summarize key features of a large body of data, and or offer a "thick description" of the data set
- Can highlight similarities and differences across the data set
- Can generate unanticipated insights
- Allows for social as well as psychological interpretations of data
- Can be useful for producing qualitative analysis suited to inform policy making

Thematic Content Analysis (TCA) approach used in this study is aimed at analytically examining the text by breaking it into relatively small units of content and "submitting them to descriptive treatment" (Sparkes, 2005). TCA involves searching and identifying dominant and unifying ideas, and common threads which extend in an interview or a set of interviews (DeSantis & Ugarriza, 2000). It allowed the researcher to produce a purely qualitative, detailed, and nuanced account of data (Braun & Clarke, 2006). The research questions posed by this study are "what" questions that are exploratory in nature (Yin 2014), TCA approach is considered a suitable approach for answering such questions, for example: "what are the concerns of people about an event? What reasons do people have for using or not using a service" (Ayres, 2007).

The chosen TCA approach, hence allowed the researchers to include the participants meaning along with the ability to develop categories. Another reason to use thematic content analysis was to identify and bring to light the overarching themes that were invoked by the participants during the interviews but remained buried in the transcripts. Similarly, a broad analysis was also carried out regarding various concepts on the frequency of different concepts based on an examination of the interview transcripts. Thematic content analysis in this study was guided by six phases (See Table 3.4), as discussed by Braun and Clarke (2006):

Phase	Description of the process
1. Familiarizing yourself with your data:	Transcribing data (if necessary), reading and re-reading the data, noting down initial ideas.
2. Generating initial codes:	Coding interesting features of the data in a systematic fashion across the entire data set, collating data relevant to each code
3. Searching for themes:	Collating codes into potential themes, gathering all data relevant to each potential theme
4. Reviewing themes:	Checking if the themes work in relation to the coded extracts (Level 1) and the entire data set (Level 2), generating a thematic 'map' of the analysis.
5. Defining and naming themes:	Ongoing analysis to refine the specifics of each theme, and the overall story the analysis tells, generating clear definitions and names for each theme.
6. Producing the report:	The final opportunity for analysis. Selection of vivid, compelling extract examples, final analysis of selected extracts, relating back of the analysis to the research question and literature, producing a scholarly report of the analysis.

Table 3. 4 Phases of Thematic Analysis

TCA technique is suitable for low level of interpretation (Vaismoradi, Turunen, & Bondas, 2013) in contrast to other qualitative analysis approaches such as grounded theory or hermeneutic phenomenology that use a higher level of interpretation and complexity.

TCA as used in this research applies minimal description to data sets, and interprets various aspects of the research topic (Braun & Clarke, 2006). TCA strategy differs from the 'pure' content analysis approach which only determines either patterns of words used, their frequency, their relationships or their structures and discourses of communication (Mayring 2000; Pope et al. 2006; Gbrich 2007). TCA was applied to minimize the risk of missing the context by using a 'pure' or 'traditional' CA approach that essentially relies on counting the frequency of codes to find significant meaning in the text.

3.5.2.2 Artefact Analysis

The artefacts collected in this research (reported in section 3.4.4) were thoroughly read, scrutinized and analysed in iterations to pick up information related to project context, initially agreed upon scope and later modifications, initial agreed high level requirements expectations from the final product with regards to functionality and quality. These artefacts were mapped with existing practices identified from interviews and observations. Such mappings helped in establishing understanding of RCM practices and an evidential link between different forms data and thus played an explicit role in data analysis adding triangulation value to this case study.

The most important use for artefacts was that they were also used to make inferences regarding the processes and practices which were then iteratively verified through further investigation and interviews (Yin, 2014, p. 107). These artefacts (especially documents) were used to "corroborate and augment evidence from other sources" as they provided specific details to validate captured information (Yin, 2014, p. 107).

3.5.2.3 Process Mapping Framework

For understanding lifecycle activities of requirements change in the studied organization their requirements engineering and change management processes were mapped. Process mapping involved constructing a model that could show the relationships between the activities, people, data and objects involved in the production of a specified output (e.g. implementation of requirements or release of software) (Biazzo, 2002). This mapping was carried out using activity based flow charting of requirements change lifecycle activities. Process mapping for requirements lifecycle activities was only after the project contract was signed between the client and the vendor in both cases.

The process mapping resulted in a framework which was later utilized for the analysis purposes of lifecycle activities of a requirements change in a globally distributed environment. Although not in the commercial software development, but a similar use of activity based flow charting of requirements engineering activities has been employed in process mapping in a GSD context by (Biazzo, 2002; Monasor et al., 2010). However, their application of this mapping was done in the education domain and it only covered the requirements engineering lifecycle and not the entire lifecycle of a requirement or requirements change (i.e. from the inception through to its implementation, testing and deployment) as done in this study.

3.5.2.4 Field notes

Notes taking was used as an aid during the transcription and analysis process. The researcher used the notes for reflection on the collected data or on the observations made to remind himself of important things written down at that point. Since all interviews were recorded heavy notes taking was not considered necessary. In some interviews the researcher was helped by a colleague in New Zealand to take down the notes as an official scribe. Since the role of scribe required only notes taking, the breadth and depth of notes were greatly enhanced. Discussions with the official scribe after the interviews also was

very beneficial for polishing the researcher's interviewing skills as the colleague was very skilled in the art of interviewing.

Notes were taken using a smart pen which provided a digital record of not only the notes as well as the time when they were taken. These field notes were quite helpful in cross checking and validating the interview transcripts. Digitally stored notes with time stamps were very useful as they pin pointed to the audio to the point when they were jotted down. It was especially helpful if at some point the audio was not clear.

The official scribe was involved only at the NZ site and only 7 interviews were conducted in the presence of the 'official' scribe because of the availability of the scribe. The facility of 'official scribe' was not available for the Pakistan site for obvious reasons (cost and logistics). Permission for audio recording the interviews was granted by all but one interviewee in case1 from Pakistan site. Notes taking became mandatory for this. Similarly, in a group meeting at New Zealand study site, where the researcher was invited as an observer and audio recording was not permissible therefore only field notes were taken.

3.5.2.5 Interview Transcription

Interview transcription was taken as a 'key phase of data analysis' of interpretative qualitative methodology as suggested by (Bird, 2005: 227) cited by Braun and Clarke 2006. The process of transcription helped the researcher to generate analytic focus which pointed out important features of data and allowed filtering out of less important ones (Gibson & Brown, 2009, p. 111). As Riessman has noted, "Analysis cannot be easily distinguished from transcription...close and repeated listening, coupled with methodical transcribing, often leads to insights that in turn shape how we choose to represent an interview narrative in our text" (Riessman 1993: 60) Cited by (Manning & Cullum-Swan, 1994)

I employed an 'unfocussed transcription" (Gibson & Brown, 2009, p. 113) approach to transcribe interview data. In contrast to focussed transcription; also known as 'narrow' transcription where the focus is on commenting and how things were said or done within the data, an unfocussed or 'broad' approach to transcription was taken to gain a holistic understanding of the collected interview data. Since the unfocussed transcription approach was not focused on a "particular section or interactional aspect of the data" (Gibson & Brown, 2009, p. 114), it provided a general overview of the entire data set rather than attempting its detailed contextual or interactional characteristics (Manning &

Cullum-Swan, 1994). This approach was not taken to just mechanically transcribe everything that was said, rather it was used to capture the richness of the data, gestures, sounds, pauses which were analysed in context later in the detailed analysis phase. Indeed, it was a time consuming and sometimes boring activity but it proved excellent for getting the researcher familiarized with the data (Riessman, 1993).

Initial reading of the interview transcripts allowed an overall impression. Later readings were meant to sift statements that were significant for the specific research questions while making use of techniques such as noting patterns, clustering and making contrasts/comparisons among data (Miles & Huberman, 1994, pp. 245-246).

In the next step data were indexed under main concepts in the research questions (manifested in interview schedule), grouped in topics to be compared with findings from previous empirical studies. The aim was on the one hand to provide a detailed 'conversational' account of patterns of technology adoption and related problematic areas, on the other hand to draw emergent themes through comparison of findings with wider empirical domain studies. Finally, interview excerpts were interpreted using thematic analysis which is one of many sub forms of CA (Sydserff & Weetman, 2002).

3.6 Validation

To establish its validity, this research has followed the set of principles proposed by Klein and Myers (1999) for the evaluation of interpretive field studies. A principle of contextualization was applied by providing rich contextual information on the social and historical background of the research setting to show the reader how the current situation under investigation emerged.

Yin (2014) notes that there are four commonly used tests to establish the credibility of social empirical research such as a case study. These tests are a) Construct validity b) Internal validity c) External Validity and d) Reliability. The most relevant categories of reliability for this exploratory and interpretive case study are the construct and external validity as internal validity is only used for the explanatory or causal studies and not for the exploratory studies (Yin 2014, p. 46). These categories are discussed in the following sections 3.6.1 and 3.6.2.

3.6.1 Construct Validity

Construct validity tries to tackle the 'subjectivity' element of the research and deals with the extent to which constructs (as operationalized in the research) relate to the research of the phenomenon under study. Yin (2014) prescribes three tactics to establish construct

validity a) using multiple data sources b) establishing chain of evidence and c) having key informants review the draft of case study report (Ibid, p. 45). Akin to the suggestions by Yin (2014), construct validity in this study is dealt with in the following way:

The questions posed to research participants in both cases followed the same theme as set out in the research protocol. As the study progressed and more interviews were conducted the participants were asked to comment on the other interviewees' opinions while keeping their anonymity intact. This served the purpose of obtaining further evidence from multiple sources on specific issues of the study. Two stage interview strategy allowed verification of previously expressed opinions by research participants but also served the purpose of member checking (verification of interpreted data) and helped in improving trustworthiness of the findings (Creswell & Miller, 2000). Data triangulation was ensured by collecting data from multiple research participants at different hierarchies in the organizations involved and from multiple sources of information such as emails, non-code artefacts and project repositories.

The help of a colleague in taking down field notes in some interviews helped in accumulating comprehensive information of the interview and further reduced the possibility of researcher's personal bias to interfere with the findings. Findings from the case study were also shared with the research participants from the concerned organizations for their review and feedback.

3.6.2 External validity

External validity or 'generalizability' requires that theories must be generalizable to the phenomenon beyond the settings in which they were generated (Gibbert, Ruigrok, & Wicki, 2008). While statistical generalizability is not possible from single or multiple case studies (Yin 1994) they do allow analytical generalization that is generalization from empirical observations to theory, rather than population (Yin, 1994). The researcher was aware of this fact hence a statistical generalization from the case study findings is not pursued.

As suggested by Benbasat, Goldstein, and Mead (1987) multiple-case designs allow for cross-case analysis however in this research each case studied is treated as a "whole" study and facts regarding the case are gathered from multiple sources as suggested by Tellis (1997).

Similarly, conclusions drawn on the facts within both cases were then used for cross comparison with the second case studied. The same study protocol was followed for data

collection and the conduct of research for each studied case to allow for replication logic as suggested by Yin (2014, p 45). Furthermore, based on the recommendations by Cook, Campbell, & Day (1979, p. 83) the context, rationale and detail for the case study selection is provided to allow the reader to elaborate on the researcher's choices. By carefully analysing the resultant data of the cases, it would be possible to form a theory from the findings, as suggested by Eisenhardt (1989).

3.6.3 Limitations

The design of the study was aimed at exploring and understanding the challenges of RCM and the associated role of collaborative technology in a GSD context, however it did have some limitations. The nature of the projects studied do not allow ready generalization to other contexts as neither of the studied projects is considered to be typical of the industry, country, or GSD practice in general. However, the knowledge shared in this study is deemed useful in other contexts via its 'thick' and rich description of the collected and analysed data. The findings presented in chapters 4, 5 and 6 in this study may enable the reader to make analytical generalization based on the research because these findings do arise from empirical data and may well be more reflective of small software companies' practices and technology use.

3.6.3.1 Other limitations of the study:

Information collected in case 1 was done predominantly from the client organization with only one interview being conducted from the vendor organization. Therefore, the identified challenges and role of CT from Case 1 organization mainly reflect the client's perspective. Similarly, information collection in case 2 was done predominantly from the vendor organization with only one interview being conducted from the proxy client; hence the challenges reflect the vendor's perspective in requirements change management. Furthermore, change implementation code and coding practices were not examined in detail. However, the findings reported in this study have drawn upon the literature, and are empirically based. Findings are triangulated by the primary author's observations of practitioners in the field, and complementary interview data to draw a stronger conclusion based on data collection from multiple sources.

Chapter 4. Analysis and Findings- Case 1

This chapter reports the analysis and findings from Case 1 of the two selected cases in this research. Section 4.1 provides the overview and background of the project and establishes the significance of the project for the client organization. In section 4.2 the global nature of the studied project is analysed using Activity Based Computing (ABC) framework. Section 4.3 and Section 4.4 provide the overview and analysis of the prescribed agile process model for the studied project in Case 1 respectively. The challenges faced due to the adoption of the prescribed model are discussed in Subsection 4.4.1 through to Subsection4.4.4.

In section 4.5 an overview of the model in practice is provided which is followed by analysis of the model in practice carried out using the lifecycle activities of a requirements change in Section 4.6. The identified challenges with each lifecycle activity are discussed in subsection 4.6.1 to 4.6.12.

Section 4.7 reports the findings regarding the complexities and challenges added due to GSD context and Section 4.8 analyses the supportive as well as hindering role of collaborative technologies used in this project. The purpose of this chapter is to present the results and findings (based on Case 1) to provide the foundation for the cross case analysis provided in Chapter 6.

4.1 CICP Project Context- Case 1

This project involves the selection, customization and deployment of a Commercial Off the Shelf (COTS) research and grant management product required by a NZ based client. The project started in 2009 and was estimated to finish in Dec 2013 with an expected cost of the project around NZD $ 3.7 million. The client and vendor teams involved in this project were geographically distributed in three countries namely, NZ, USA and Canada.

The client is a medium-sized, non-profit, education and research organization with around 3,000 staff. The vendor is a US-based SME trading for approximately six years (with a small team of five to six people), that offers a COTS based research and grant management product for the education industry.

Before the start of CICP project, the client organization carried out evaluation of some of the leading commercially available research and management solutions. Finally, they selected a vendor and went into a contract for the procurement and further customization of their product. The client reasoned that buying an off the shelf commercial package

would be more cost and time effective as compared to modifying the in-house developed software or building a similar application from the scratch.

The CICP project was considered suitable for this study because it falls under GSD's *offshore outsourcing* category (i.e. collaboration of different organizations in different countries and "leveraging external third-party resources situated in a different country") (Smite et al., 2014). Subsections 4.1.1and 4.1.2 establish the need of the project and provide the contextual factors respectively. For qualitative research, studying the context is important because it helps to understand the relationship among social groups, individuals and artefacts Nardi (1996). Runeson and Höst (2009) recommend to include contextual details of case studies to improve the standard of reporting. Providing ample detail on the case study context is also considered desirable because it strengthens the external validity of the study (Gibbert et al., 2008) and allows the readers to appreciate the sampling choices made by the researcher (Cook et al., 1979, p. 83).

4.1.1 Need for the Project

The two factors that made the client look for a new research and grants management system were a) the growth of the research and research funding in the client organization and b) the inability of the existing in-house built software to manage demands imposed by such growth. The new system was to manage information and risks associated with $25M of the organization's annual revenue and $40M of research funding[6]. With the growth in research and the available grant money came the additional demands on the management, storage and reporting of various research activities and outputs. This also meant efficiently managing the whole process and producing informative reports for the organization's researchers, managers and administrators became even more important. These demands could not be met by the existing in house built system. Moreover the existing manual process for ethics approval was considered inefficient with limited ability to share the required paperwork among the concerned parties[7]. Therefore the business case for the new research and grant management system identified three main areas for support [6] namely 1) research activities and outputs 2) internal and external research grants and 3) ethics approval process.

4.1.2 Contextual Factors of the Project

A summary of various contextual factors of CICP as a globally distributed development project is provided in Table 4.1. Petersen and Wohlin (2009) consider this type of

[6]Project Plan Statement of Scope Version 1.3, 7th Sept 2011
[7] Project Vision Draft 09 Nov 2009

contextual information to be useful in summarizing the industrial context of GSD projects. They argue that such information is helpful in evidence aggregation for GSE studies and aids the researchers in making informed and meaningful decisions about GSD studies. Table 4.1 extends the contextual factors proposed by Šmite, Wohlin, Gorschek, and Feldt (2009), as well as those proposed by Petersen and Wohlin (2009) to include collaboration technology a notably missing element in both listings.

Factors and Dimensions	Detail
Market	
Industry	Education
Sector	Commercial
Product	
Maturity	Custom Development/COTS Extension
Software	Research Management System
Application Type	Web Application
Size	Small
Cost	NZD $3.7 Million
Complexity	Medium
Requirements Change Rate (Volatility)	Low to Medium
Development Organization / Vendor	
Maturity/Certification	SME (Contracting since 2008)
Team Size	6
Development Process	
Outsourcing	Offshore Outsourcing
Methodology	Hybrid: (Agile-Waterfall)
Workflow	Scrum Based
Practices	User Stories, Time Boxing
Global Collaboration	
Collaborating Units	NZ, USA, Canada
Collaboration Technologies and Tools	Email, Skype, JIRA, Spreadsheets, GoTo Meetings
Collaboration Artefacts	Product Backlog, Sprint Backlogs, High Level Req. Specifications, Issue Logs, Software Releases
People	
Client Collaboration Roles	Project Manager, Product Owner, Business Analysts, Tester
Client Governance Roles	Product Owner, Project Manager, Steering Committee
Vendor Collaboration Roles	Vendor BA/Architect/ Principal, Lead Developer, Software Developers
Vendor Governance Roles	BA/Architect/ Principal
Project	

[8]Status	Finished: Product delivered to the client
Outcome	Failure- Falling short of promised features and user benefits. Product considered of limited use by the client due to: • Low confidence in the reliability • with a very slow processing speed • frequent urgent remedial work required to specific events

Table 4.1 Project Context- Software Engineering Adapted From Hussain & Clear (2014)

In the next section 4.2, a contextual analysis of the CICP project is provided based on Activity Based Computing Framework (Korpela, Mursu, & Soriyan, 2002). Using various concepts of ABC framework, various sites, actors, artefacts and collaborative technologies used in CICP project as well as the interactions of these contextual elements are depicted and analysed. This contextual analysis of the CICP project is aimed at presenting the CICP project context in a meaningful way and preparing the ground for comparative understanding and analysis across the two studied cases.

4.2 Contextual Analysis of the CICP Project Using ABC Framework

The contextual depiction and analysis of the CICP project is based on Activity Based Computing (ABC) Framework provided by (Korpela et al., 2002). According to Korpela et al (2002), an activity is defined by "the shared object and its transformation into the jointly produced outcome". The purpose of an activity is to transform a shared object into an outcome which is achieved through a methodical interaction of various elements involved in the work process. ABC framework is used here for presenting software development as a real life work activity. It is aimed at providing an appropriate analytical abstraction that discards irrelevant details but at the same time isolates and emphasizes the most significant properties of artefacts and situations (Brooks, 1991).

Activity theory is frequently used in understanding and reflecting on software development (Bodker, 1997; Nardi, 1996). Korpela et al, (2002) applied the ABC framework to understand and analyse information system development as a practice of 'real-life *work activity* in context' (2002). de Souza and Redmiles (2003) have used activity theory to study large scale software development collaborative work. In global

8Status reported here is subsequent to the study i.e., at the time of finalizing the thesis. Sources of status information: 1. Benefit Realization Report, which also noted ([one] module was not implemented as management decided that the intended benefits would not be realised through the implementation of a further module) 2. Personal communication with one of the project steering committee members.

software development Tell and Babar (2012) applied the principles of ABC framework to understand system activities and model development processes such as software architecture design and architecture evaluation processes.

In this research ABC framework is considered useful because it can allow researchers to analyse their work and identify the areas requiring improvement ('misfits') (Korpela et al., 2002). ABC framework provided the research with a mechanism to present a synthesized an abstracted view of the studied project which is used for further analysis of the case under study.

4.2.1 Adaptation of ABC Framework for the CICP Project

The ABC framework used in this study is adapted and enhanced based on the visual notation taxonomy proposed by Laurent et al. (2010). The activity based model presented by (Korpela et al., 2002) (see Figure 4.1) is enhanced by adding visual notations from Laurent et al. (2010) as the model used by Korpela et al. (2002) does not contain any standard notation for work activity presentation. There are two points of difference in the application of the framework in this study 1) its adaptation and application in a GSD environment across multiple sites; an idea implicitly suggested in Korpela et al. (2002) work of studying ISD as a 'boundary-crossing' practice and 2) addition of some visual notations and symbols to aid visualization of requirements engineering in a distributed context. The added symbols are based on the visual notation taxonomy applied in modelling globally distributed requirements engineering processes by (Laurent et al., 2010). They claim that it aided project communication between the team members and practitioners. The notation is considered useful in this study because it helps in planning, analysing and optimizing distributed requirements engineering processes (Laurent et al., 2010).

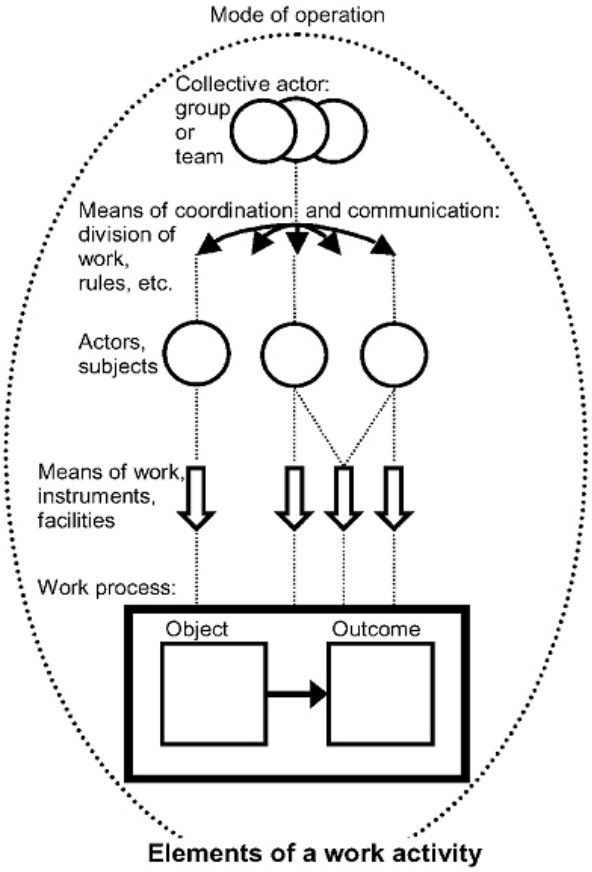

Figure 4. 1 Elements of Work Activity Model (Korpela et al. 2002)

The model shown Figure 4.7 (page 84), is enhanced by adding the iteration symbol and the actual activities involved in converting a shared object (such as a requirements specification) into an outcome (a release) which was missing from the model. The model is further enhanced by breaking up the abstraction of "*mediating work instruments*" and "*work elements*" into mediating artefacts, tools (Figure 4.7).

In adopting the visual notation of (Laurent et al., 2010) the three basic elements presented in the model are roles, sites and artefacts. Stakeholder roles are depicted as human shapes shown as one, few, or many stakeholders. Sites are depicted as coloured boxes. The technologies for communication and collaboration are depicted by commonly understood technology symbols or the icons of the actual product used in the project. Artefacts are depicted through well-recognized symbols used in software engineering and development. Relationships are shown by means of arcs: a solid line represents co-located communication between roles, a dashed line represents distributed communication between roles, and a dotted line represents the relationship between a role and an artefact (See Figure 4.2).

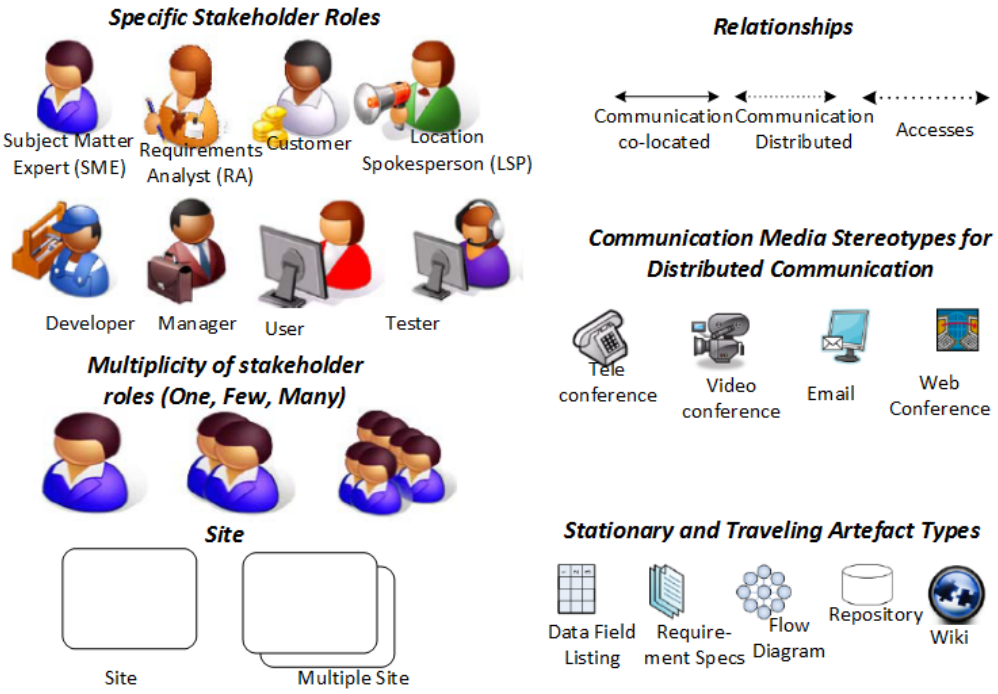

Figure 4. 2 Visual Notation for Modelling Globally Distributed Requirements Engineering Processes by Laurent et al 2010

The notation proposed by Laurent et al. (2010) is itself enhanced by a) adding icons for prototypes or mock-ups, (missing in the notation), b) adding relationship arcs for onshore collaboration shown through relatively more strongly dotted lines and c) specifying all the individual sites and their roles, compared to the concept of 'multi-sites' shown as multiple boxes which hides the actual stakeholders of the sites involved (Laurent et al., 2010), d) removing site and time difference information from the site boxes and placing them at a convenient position for clarity and readability by using arrows and labels pointing to the actual site, and e) adding onshore relationship to the existing co-located and distributed relationships used for communication among roles by using stronger dotted lines than the ones used for distributed communication (See Figure 4. 4). The following sections discuss various elements relevant to the converting of the design specifications work activity work activity into a production release.

The terminologies and concepts of Activity Theory provided a convenient way to depict and model various stages and elements of requirements (or change) implementation as a *work activity*. The visual depiction may help in understanding existing processes, identifying and highlighting weaknesses or problems. It may also help in establishing improved processes and supporting infrastructure that are suitable in a GSD context (Laurent et al., 2010). They are aimed at aiding the reader with a mechanism to grasp the elements of a work activity in a single snapshot.

Another reason for adopting the notation, as suggested by (Laurent et al., 2010), *w*as its potential to provide *"a common language for modelling distributed requirements projects and activities, and thereby facilitates comparisons across projects. These comparisons make it possible to identify recurring patterns of collaboration, common obstacles, and best practices used for collaborative requirements engineering activities. Such observations enable researchers to propose new techniques or improve existing methods to handle the specific challenges of global requirements processes"*

4.2.2 Mapping CICP Work Activity Elements Using ABC Framework

An example activity from the CICP project i.e., converting sprint requirements into a software release, is mapped on to ABC framework using its various concepts and work elements. The elements of Activity Theory (Korpela et al., 2002) mapped to the conversion of sprint requirements into a software release as a work activity include: *Object, Outcome, Actors (and Sites), Means of Communication and Coordination, Means of Work / Mediating Instruments: Mode of Operation*. Mapping of work activity elements is done on the basis of information collected from the interview data and the available artefacts. Mapping of concepts such as '*means of work*' and '*mediating instruments*' and *'mode of operation'* is based on the analysis of project wiki, bug tracking system and other project related artefacts such as requirements specification spreadsheets, email excerpts, screenshots, design specification models) stored on these software applications.

The work activity and its elements (roles, sites, means of communication and collaboration) synthesized from the collected data are presented here using an abstract visualization for a broader understanding and analysis of the project work (See Figure 4.7 page 84). This project level view complements a detailed analysis of requirements and change lifecycle activities is carried out later in Section 4.6 later.

4.2.2.1 Object and Outcome

For CICP project an example work activity is mapped using ABC framework which takes sprint requirements as a shared *object* for this activity. Through collaboration of people (a collective activity) following a predefined process this shared object is converted into an outcome i.e. a software release in this case.

4.2.2.2 Mode of Operation

The mode of operation for carrying out this shared activity is both Collaboration driven as well as Technology Driven. It means that the shared object is converted into and outcome through collaboration carried out using various tools and technologies.

4.2.2.3 Actors and Sites (GSD Context)

This globally distributed development project was carried out between a client organization which is located in NZ and the US based development organization. The client and the vendor sites have almost eighteen-hour time difference. There is a slight variation in the language used (English) in these countries. Similarly, there are cultural differences between the countries involved discussed later in the analysis. There is no prior history of working together of the client management team and the vendor's team. All activities related to requirements communication, negotiation and validation are coordinated over geographic, temporal and cultural distance. In this project there were six vendor sites in total out of which five were in USA and one was in Canada. There was only one client site in New Zealand which is shown in the left side of Figure 4.3.

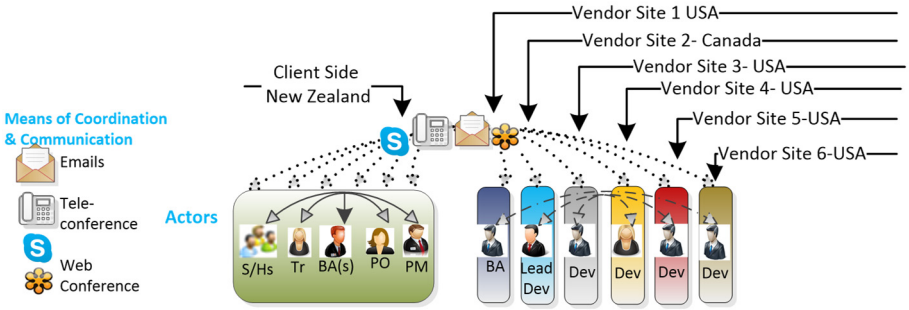

Figure 4. 3 Actors and Sites Involved in Distributed Collaboration

The actors involved in this work activity from the client team included a Project Manager (PM), a Product Owner (PO), three Business Analysts (BAs), a tester (Tr) and other stakeholders such as a steering committee (S/Hs), who were all based at the client's site in NZ. The project on the client side was also governed by a steering committee which mainly consisted of key stakeholders from the senior management. The vendor team actors involved in this activity were the Business Analyst / Architect and five geographically distributed developers working from various locations in Canada and the USA (Figure 4.3). Actors on both sides apply domain and contractual knowledge to convert requirements and changes into a release by applying several means of communication & collaboration and utilizing various mediating instruments.

4.2.2.4 Means of Communication and Coordination

Various communication and coordination means were applied to carry out the collaborative work however the usage differed within and across sites. Asynchronous communications were mainly carried out through emails to communicate requirements

and changes and collaboration over requirements related activities. A project wiki is also used for asynchronous communication but it was mainly used locally by the client team members (See Figure 4.4).

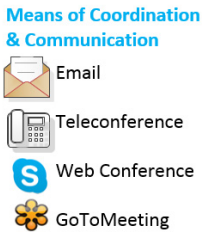

Figure 4. 4 Mediating of Coordination and Communication Used by Distributed Actors

Synchronous communication channels for communication (Beecham and Noll, 2010) as well as collaborating for requirements are *GoTo Meetings* and *Skype*.

Artefact sharing and exchanges are done both through asynchronous and synchronous means (along with the screen sharing feature of the software). These activities are meant to support activities of requirements elicitation, understanding, clarification and negotiation. Exchanging emails, spreadsheets, mock-ups and other requirements related electronic artefacts were common for both within and across site communication and collaboration during requirements implementation activities.

Face to face meetings were only used for within site whereas phone calls were meant to facilitate cross site communication and collaboration. Similarly, offshore visits were applicable only when the vendor or his team paid a visit to the client.

4.2.2.5 Means of Work-Mediating Instruments

The work activities in Figure 4.5 are carried out in a collaborative and technology driven mode. Vendor and client team members implement customization requirements into relevant sprints utilizing various instruments and means of work. The tools used in this activity included graphic editors (such as MS paint), modelling software (MS Visio), Spreadsheets (MS Excel), Emails, Project Wiki, Software Release[9]. Here are some of the examples of the use of the means of work and mediating instruments (See Figure 4.5).

[9] Project Wiki *Confluence*, *JIRA*

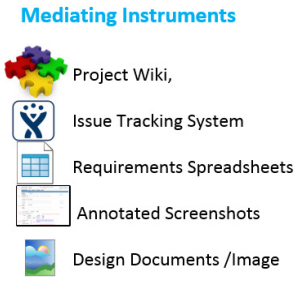

Figure 4. 5 Mediating Instruments Used by Actors

In collaborating with the client team, the vendor development team finalizes the requirements specified in *Excel* sheet templates (containing high and low level requirement specifications), workflow diagrams and annotated screenshots (Hussain et al 2014). Requirements related artefacts themselves act as means of work and mediating instruments which are produced by other instruments such as MS paint (for annotated screenshots), MS Visio (for making work process diagrams). Once implemented, the requirements are released into a new version (release) which is deployed on the client environment where client BAs and tester perform testing and report issues in the project collaborative technology/wiki (*Confluence*) (Atlassian, 2015). The issues identified are communicated utilizing the same communication means and channels used to share requirements with the vendor development team.

The cyclic activities in this transformation include: 1) Implement, 2) Test / Retest 3) Bug Fix 4) User Acceptance and 5) deployment of release (Figure 4.6). Within the big iteration of the six steps mentioned here activities 2 and 3 undergo a number of small iterations until the release is stable enough to be presented to the users for their acceptance.

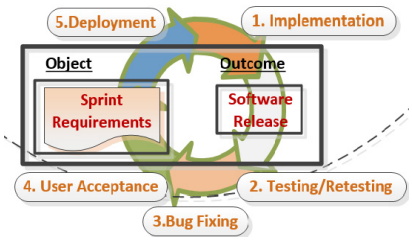

Figure 4. 6 Conversion of Object (Sprint Requirements) into Outcome (Software Release)

Finally, Figure 4.7 depicts the globally distributed nature and all the elements of the work activity for converting sprint requirements specifications into a software release in one snapshot. The actors are shown using icons with their roles enclosed in coloured boxes specifying the geographically distributed sites they belong to. These actors perform various activities based on the assigned roles and the division of labour utilizing the Means of Coordination and Communication and the Mediating Instruments shown in

Figure 4.7, and are explained (using icons) on the left hand side of Figure 4.7. Collaboration among the actors is shown by arrows and curved lines (dots suggesting weaker collaboration links). The work process transforms Object into Outcome, which in this example is iteratively transforming Sprint requirements into a software release (See Figure 4.7).

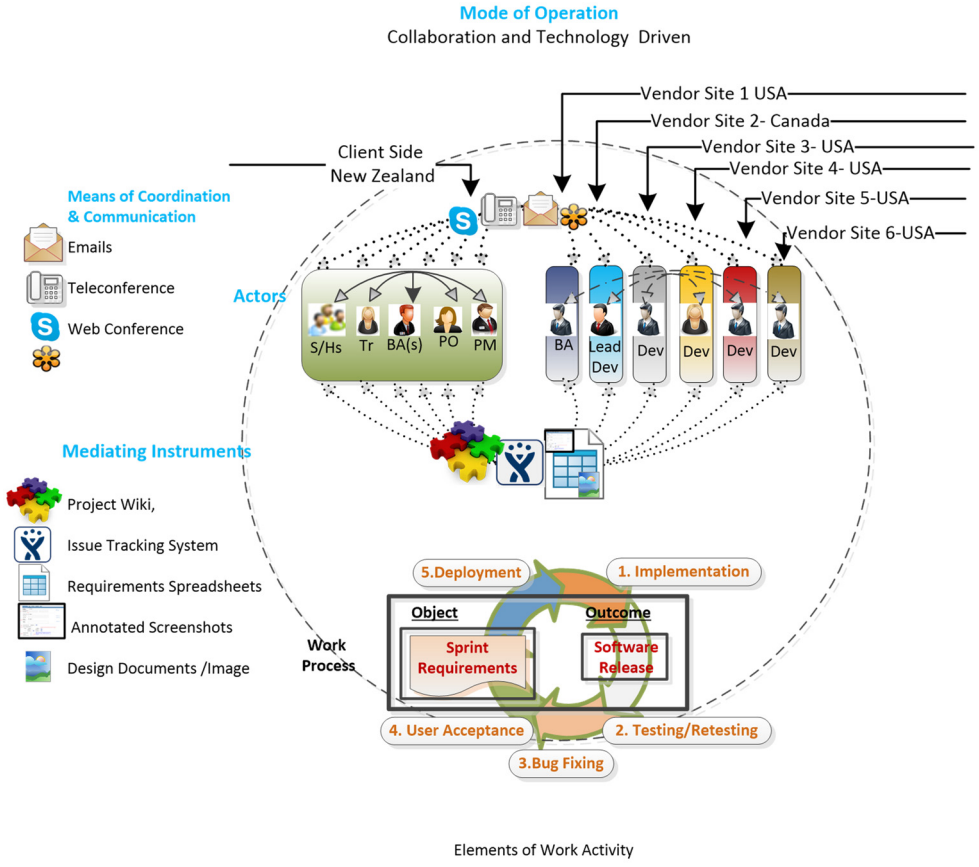

Figure 4. 7 ABC Based Model of the CICP Project Showing Elements of Work Activity

4.3 Overview of the CICP Prescribed Process Model

CICP is a customization intensive (COTS) product development and integration project, carried out for one particular client with geographically distributed stakeholder groups. The prescribed process model shown in Figure 4.8 was instituted and advocated by the client organization as a guide for project development and management to be followed by both the client and vendor teams. Since the model is based on agile methodology it provides a basic guiding structure of the project development as change management. The model provides definitions of the various elements included in the process which include (among others) Product Backlog, Iteration Backlog, Execution and Iteration Review Meetings and other practices such as user acceptance testing burn down charts

and daily stand-up meetings. The roles involved are also named and given a brief description according to their responsibilities in the process see Figure 4.8.

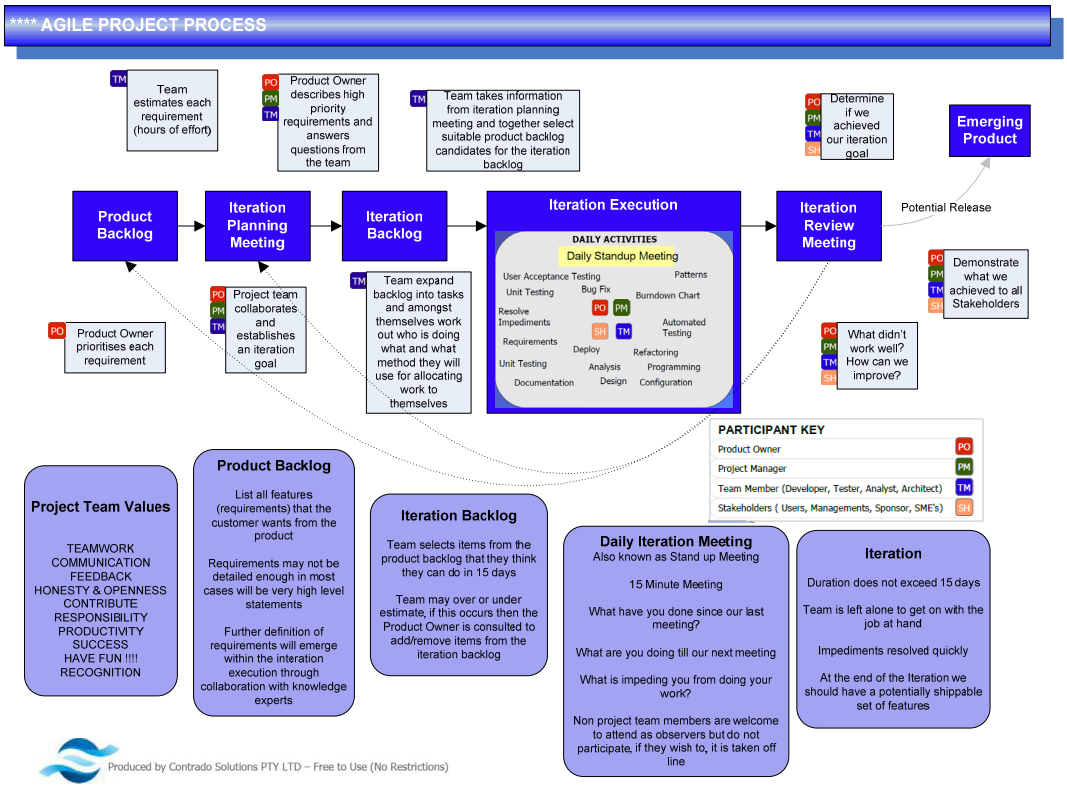

Figure 4. 8 Prescribed Agile Development Process for CICP Project [10]

According to the client they followed the basic practices prescribed by this agile-like process model. The client described some of the process activities from requirements gathering through to the development as follows:

FRNZ-ICK-05-2013 "I would describe it as Agile, we work with a stakeholder group to write stories which document the changes that need to be made to the packaged software and then those stories are used by our business analysts to write the specification which documents in more detail the actual changes, field by field level, that need to be made to the application. It is that specification with the field by field changes that goes to the vendor, who needs to make changes. So that's basically requirements gathering [and] specification. Vendor takes the specifications they might have questions for us there might be a teleconference to walk through the specifications and deal with any shortage of understanding and then basically they may take that and do the development."

According to the vendor the project methodology in CICP project's related to agile and the development was carried out iteratively, using the notion of sprints.

FRNZ-AVA-06-2013 "in our contract there is a notion of a backlog, it is following agile methodology…a sprint structure, each of the sprints were named and had a high level scope assigned to them, relative to the phase of work, [which] would be specific to the domain need."

[10] Project Wiki Background Reading [Project Name] Agile Project Process v 0.2.pdf

The vendor considered the methodology to be a 'hybrid' of agile and waterfall as it followed elements of both

FRNZ-AVA-06-2013 "there was no five-hundred-page waterfall document pre project, this was a hybrid agile waterfall so it presumes the customer knows the business and they come prepared with the right staff and it is all supposed to work relative to a baseline."

Requirements elicitation, analysis, implementation and testing were each time-boxed in sequential two-week "sprints". This resulted in a six-week development iteration length. The approach was viewed as "agile" by the client team since it was iterative and some management techniques used by the client are commonly associated with agile methods (e.g. user stories, product backlog). The client commented on the methodology and the concept of time bound application development chosen for the project as under

FRNZ-ICK-05-2013 "I have heard people talk about hybrid agile waterfall style, I guess, [the methodology in the project] is agile and we are really we are constraining the period during which a certain number of things will get done to deliver something that is useable but within a six weeks process [sprint/Iteration]".

4.4 Analysis of the CICP Prescribed Process Model

The analysis of the prescribed CICP process model and the interview data revealed that the prescribed model did not reflect the actual practices carried out by the participants. There were several concepts and practices in the prescribed methodology that were related to Scrum. However, in practices these concepts and related activities were not followed. For example, the vendor team did not work with user stories and required the client to provide requirements specified (in a greater detail as compared to user stories) in vendor-provided spreadsheet templates. Similarly, many other 'Scrum' practices were not followed such as daily stand-up meetings, iteration planning and iteration review meetings as suggested by prescribed model.

The vendor team was also not working with the scrum prescribed roles such as Scrum master and product owner in their development team. Furthermore, the contract that required several high level modules to be modified which served as the product back log and the requirements were not prioritized. Chunks of functionality from the contract (and its high level modules) were selected by the client and vendor to be developed in a certain time which was called an iteration but the iteration time frame varied initially from three weeks to six weeks in the later stages of the project. Furthermore, the prescribed model was not adapted to suit the globally distributed nature of the project and hence did not

have practices and roles specific to a GSD context. The identified challenges in the application of the prescribed process model are described here:

4.4.1 Methodology Misalignment and Disagreement

The client and vendor did not agree on the project methodology and had their own methodology preferences. A pure agile methodology (based on Scrum) for the project development and management was proposed by the client which the vendor did not agree to. An iterative project development approach had to be negotiated with the vendor team who preferred a more controlled and waterfall like methodology.

FRNZ-AVA-06-2013 "I pushed very very hard to make sure that things were a little bit controlled. We ultimately agreed on a sort of hybrid agile waterfall methodology, again the waterfall aspect being the time you take to be thoughtful about what you are doing before you do it, not just reacting all the time".

The model shown in Figure 4.8 is shared by one of the client BAs from the project wiki when asked to verify a process model drawn by the researcher which covered requirements engineering and implementation activities as well as the overall project workflow. Following is the email conversation regarding that artefact.

29/05/2013 "As discussed, here is the sketch of the project workflow which I have tried to draft based on the information from the interviews so far. I would appreciate your input to improve my understanding of the process and workflow."

*11/06/2013 "The diagram you sent through isn't quite right and it would take me a bit of time to correct. Are you able to access the following link: https://********. It outlines the standard Agile process we are following".*

There was however, a disagreement regarding the adopted methodology for the project among the client and vendor team members. The vendor considered that the methodology suggested by the client was not suitable for the project and strongly advocated for modifying the model and bringing it more close to a waterfall / agile hybrid one.

FRNZ-AVA-06-2013 "Agile methodology by itself tends towards very rapidly developed but horribly 'siloed' applications… you are focusing more on speed of solution delivery versus thoughtful integration to the enterprise"

From the vendor's perspective a plan driven methodology such as waterfall could provide more control over the development activities of the project. The vendor questioned the client preference of using a pure agile methodology in the project and took a strong stance against it.

FRNZ-AVA-06-2013 "honestly I would say that [the client] had a bias towards their own methods. …. I got many serious discussion with [the client] project management and then I would escalate…I just said look if you want us to take responsibility for the ultimate deliverables then you will follow the

methodologies as we discussed or we will stop working and leave the contract with [the client organization].

Both parties eventually agreed on a modified, iterative and time boxed development process.

4.4.2 Lack of Application of the Prescribed Model

A close analysis of the prescribed agile process model applied in this project revealed that the model (in Figure 4.8) was not tailored for geographically distributed development. It did not capture a) the multiple (client and vendor) sites involved b) cross site team members, c) site-specific roles d) cross site interactions and e) collaboration mechanisms of the team members.

The applicability of the model as a common process was difficult as it did not fit well with the day to day project activities of the cross site client and vendor team members. Furthermore, it was difficult to ascertain and thus map the sequence and timing of the prescribed activities in the model to the actual practices of the two organizations involved. Major deviations in practice from the prescribed process model itself renders the model of little use for the team members. For example, among the *Daily Activities* described in the "Iteration Execution Process", the *Stand-up Meetings* were not performed which the client thought was due to the geographical distance between the client and the vendor" (See Figure 4.8).

The project wiki records only two of daily meetings[21] and one of the client stakeholders confirmed that daily meetings were abandoned after the initial few meetings. The client and vendor teams did not even have a regular weekly meeting.

FRNZ-CKE-05-2013 "I tried to set up a weekly meeting with the vendor, they did not want that, they said we only want to talk to you if you have got something specific to discuss. I would have preferred a placeholder every week, just to say this is what we are doing this week. To get like a regular update session".

The notion of a 'team' in the prescribed model was not clear hence it was difficult to identify whether the prescribed process was meant for the client or the vendor team or both. Such a lack of applicability of an untailored and 'purely agile' prescribed process model in the CICP project made one team member from the client side question its usefulness in GSD projects.

The client team itself considered the methodology was not closely aligned to the prescribed agile methodology as shown in the prescribed model in Figure 4.8

FRNZ-CKE-05-2013 "I think we are doing agile but it is not very agile, because we haven't had a release in a month which you wouldn't consider agile. We would love more regular releases we would love more regular contact. We have tried a few different ways of engaging with them and they seem to go underground and pop up and deliver something…not very agile at all"

4.4.3 Lack of Codification of the Agreed Methodology

The prescribed process model (Figure 4.8) went through periodic modifications from pure agile to "hybrid agile-waterfall". The modified versions of the agreed process model were not codified (written down) and the only codification of the guiding process model remained the same as shown in Figure 4.8.

Similarly, the change management process followed in the process was not written down. It was assumed that the prescribed process model (Figure 4.8) and the experience of the people involved would enable them to understand and follow a common change management process. Hence the need was not felt to codify the process.

FRNZ-ICK-05-2013: "We haven't codified the [requirements change] process to the extent that we have written it down but I would say that it is a fairly standard practice and people who are experienced at working on packaged software implementation or software development projects understand this process. So I don't think it needs to be written down".

In fact, the project did not have a formal change management process at all. Again the perception was that the prescribed agile process model (Figure 4.8) will enable the practitioners to manage requirements changes based on scrum methodology practices.

FRNZ-CKE-05-2013 "we don't have the formal change control at this stage, any changes that come, as we are supposedly working in an agile fashion, so they just come off the backlog and they get assessed like any other piece of work"

4.4.4 Inadequate Requirements Change Management Process

It is important to note that the CICP project's change management plan was codified and in place[11] but requirements change management process was not. The change management plan stated that changes having a ten percent or more impact on the project cost were to be channelled formally through the change management process. For the changes with less than ten percent impact the product owner had the authority to make the final decision. Most of the changes in requirements had less than ten percent impact on the project budget and therefore were decided by the product owner.

FRNZ-ICK-05-2013 "each change needs to be discussed with the product owner to get final approval… the product owner has to rely on the [client] Project Manager and the business analysts as to what the

[11] Project Execution Document v 1.0P included in Phase 2 Statement of Scope V 1.4 16 09 2014

impact change involves. The product owner should be able to weigh up the benefits of that change versus other things that might have to be removed from the scope to allow that change to happen….

As per one of the BAs the product owner relied on the project manager and the BAs to ascertain the impact of making a decision regarding change inclusion in the project. However, in reality the product owner had to rely on the vendor's product knowledge and expertise to ascertain the impact, required effort and price involved in making a change. In the absence of a formal codification of the change management process the client had no control over how change was processed or managed and had to rely on the vendor's decision on changes were communicated or handled. According to one of the client team members, changes were negotiated via a teleconference and the impact was documented usually in emails without any particular format.

FRNZ-ICK-05-2013 We [the client team] [would]certainly be relying on the developers to tell us in terms of what the impact, the complexity, time and cost would be for that change…it would probably be via an email conversation we would certainly get it in writing before we proceed."

In summary, the prescribed process which was meant to aid the understanding and sequencing of the activities in the project actually caused confusion and conflict. Methodology preferences, its lack of applicability and GSD related challenges added to the confusion and hence the practices described in the model were not actually followed.

4.5 Overview of the CICP Requirements and Change Process Model in Practice

From the analysis of the project data (collected from interviews, electronic artefacts and email excerpts) the actual sequence of requirements and change related activities as practiced started to emerge. Thus, an empirical model to capture the actual requirements and change related practices was constructed (Figure 4.9). It presents the activities and their sequence that is 'closer to the actual practice' in the CICP project. The CICP model in practice can be divided into three main activities 1) RE activities 2) Implementation and 3) Testing and Bug Fixing (Figure 4.9). This research considers that when a requirement (or a change) follows all of the aforementioned three activities, it constitutes a complete lifecycle. Furthermore, this research considers that both requirements and changes may go through the same lifecycle activities and present the same challenges for the people responsible for managing them. This concept is in line with (Ambler, 2014) who suggests that "any change to a requirement you're currently implementing should be treated as just another new requirement" and therefore a change should follow the same lifecycle activities as that of a requirement.

The RE activities covered in the CICP model in practice (Figure 4.9) can be mapped on to the RE process activities described by (Sommerville & Sawyer, 1997, p. 11) which are a) *Requirements Elicitation* b) *Requirements Analysis and Negotiation*, c) *Requirements Documentation / Specification* d) *Requirements Validation*. For the purpose of this thesis (as noted earlier in Chapter 1 pg. 1), these main lifecycle activities (LCAs) to manage requirements change are extended to include *Prioritization*, *Managing Uncertainty*, *Traceability*, *Implementation* and *Testing* based on ISO/IEC TR 24766:2009 framework (de Gea, Nicolás, Fernández Alemán, Toval, Ebert, & Vizcaíno, 2012).

The challenges to managing requirements were identified inductively from the interview transcripts and were categorized using thematic content analysis technique as discussed previously in chapter 3.2.3. The boundary of these process activities were not as strictly defined in the studied project as suggested by (Sawyer, Sommerville, & Viller, 1997).

During the RE process several requirements related activities went in parallel. For example, analysis, specification or validation were being performed almost alongside Requirements Elicitation, in an iterative manner as discussed by (Zowghi & Coulin, 2005).

The input to the CICP model in practice (Figure 4.9) comes from the 'conceptual' product backlog which was composed of high level requirement statements in the contract outlining major areas for product development or modification. A high level sprint scope was taken from this conceptual backlog. For each sprint all three main activities 1) RE activities 2) Implementation and 3) Testing and Bug Fixing were followed as shown in (Figure 4.9).

In the CICP model *Requirements Elicitation* activities dealt with identifying, analysing and recording candidate customization requirements from various sources. *Analysis and Negotiation* activities dealt with detailed analysis of requirements and negotiation among different stakeholders to decide and prioritize which requirements were to be accepted for each sprint. *Requirements Documentation / Specification* activities involved detailing and documenting requirements. These specifications were based on vendor negotiation for scope, cost and complexity or the requirements / customizations to be implemented. *Requirements Validation* involved validating customizations or requirements with the client as well as performing analysis and negotiation with the vendor. It also included careful checking for the consistency of the specifications previously detailed. *Implementation* activities covered implementation of customizations and requirements by the vendor's development team using time boxed sprints. The outcome of each sprint

went through *Testing and Bug Fixing* activities. To provide stability during sprint requirements were 'frozen' for each sprint as suggested by 'Scrum' methodology (Ambler, 2014).

The lifecycle activities of a requirement or change in requirements as observed in CICP project are captured in the model in practice (Figure 4.9) and discussed in detail through the model in the coming section 4.6.

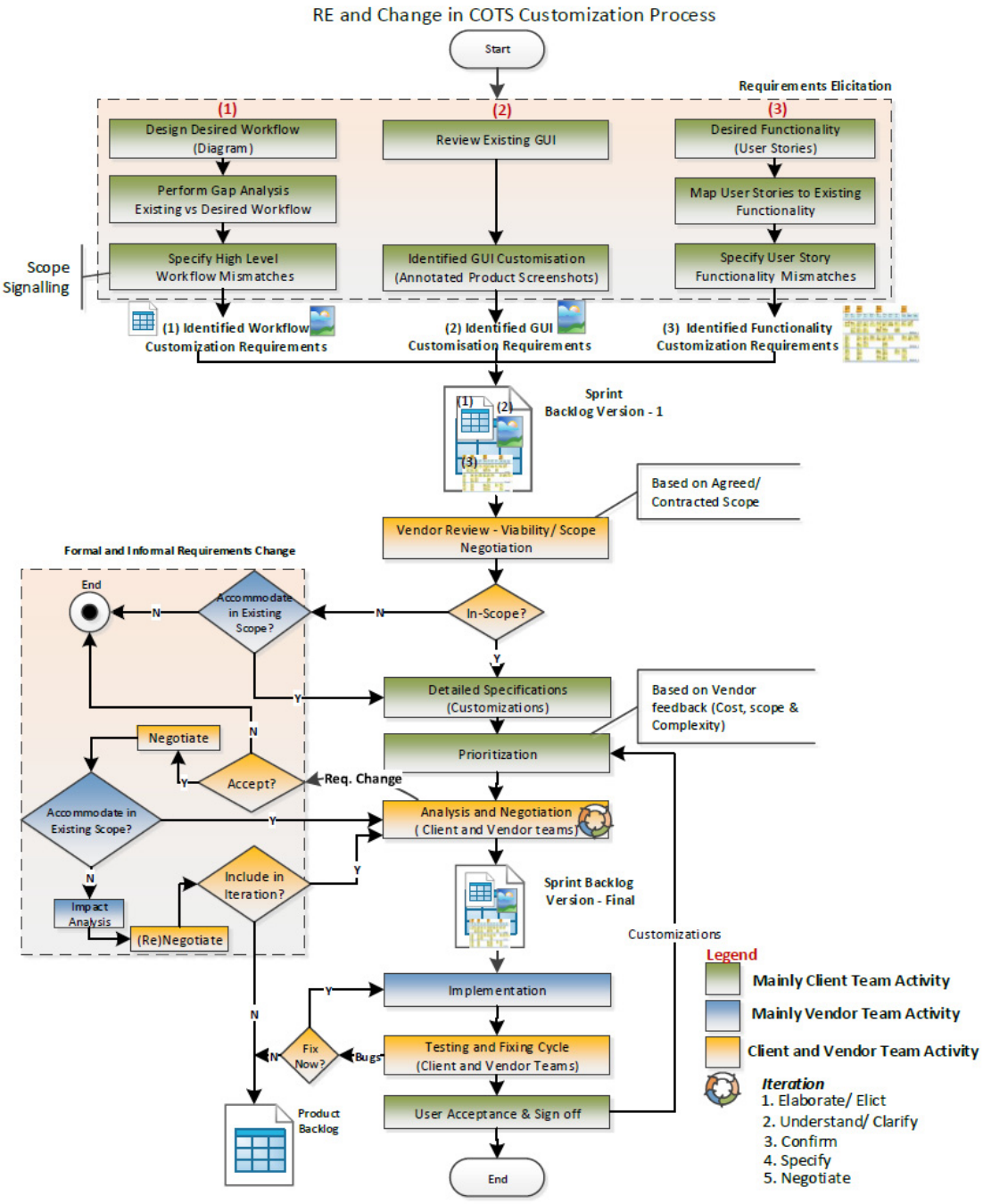

Figure 4. 9 RE and Change Process Model in Practice (Hussain et al., 2014)

For the purpose of simplicity and close analysis the model is divided into six sections (see Figures 4.10-4.17) based on the complete lifecycle activities discussed previously. The lifecycle activities of a change and the associated challenges were analysed in detail using the CICP model in practice in the coming sections.

4.6 Analysis of the CICP Requirements and Change Process Model in Practice

In CICP elicitation and its related challenges are discussed as the starting point of the requirements change lifecycle. This is because the literature on requirements management in both the traditional (Davis, 2013; Leffingwell & Widrig, 2003) as well as agile methodologies (Karlsson, Dahlstedt, Regnell, Natt och Dag, & Persson, 2007; Livermore, 2008) identify elicitation as the initial activity in change management lifecycle.

4.6.1 LCA1: Requirements Elicitation

The three main sources of elicitation for the desired customization requirements and changes in the CICP project were a) the existing and desired workflows b) GUI of the existing product and c) user requirements. These three work streams were undertaken by the three client BAs and are discussed below.

4.6.1.1 Stream 1: Desired Workflows

Requirements elicitation in this stream was carried out by comparing existing process and product workflows with the desired workflows and then specifying high level workflow mismatches. In order to ascertain customizations detailed workflow diagrams of ‘as is’ and ‘to be’ processes were developed based on the information from key stakeholders. The workflows which were missing or identified as inadequate in the product represented candidate customization.

4.6.1.2 Stream2: GUI Customizations

This was done by one of the client BAs by reviewing the existing product GUI and identifying the enhancements and desired changes. Sometimes these customizations were related to USA-specific GUI aspects of the COTS product that did not suit the NZ context. This resulted in a set of annotated screen shots representing candidate COTS customizations.

4.6.1.3 Stream 3: Desired Functionality-User Requirements

The functionality desired by the users in the form of user stories was compared with the existing features offered by the COTS product. The mismatches of functionality were identified as customization requirements (See Figure 4.10). The user stories for customization requirements were then analyzed, specified and prioritized locally before sending them to the vendor for his review.

FRNZ-CKE-05-2013 "We don't formally do a gap analysis, but in effect that's what we are doing, we are looking at the product to see what enhancement we need to do to basically meet the stories that the business (people) have requested".

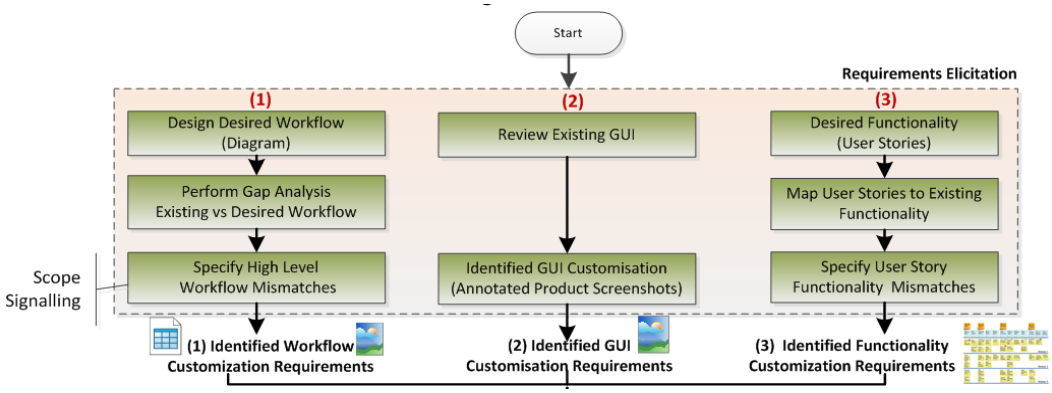

Figure 4. 10 Model in Practice-Elicitation and Initial Specification Section

4.6.2 LCA 1 Requirements Elicitation -- Challenges

Elicitation was one of the most important and most challenging areas in the CICP project which had ripple effects on almost all other change management activities. The elicitation challenges in CICP related to 1) Separation of Requirements from Requirements Change, 2) Client Led Elicitation 3) Lack of Product Knowledge for Feature Identification, 4) Lack of Domain Knowledge for Product Internationalization, 5) Lack of Product Documentation, 6) Lack of availability of the Vendor's Expert Product Knowledge, 7) Conceptual Distance between Requirements and Product Features 8) New and Partially Developed Product 9) Vendor's Dominance and Credibility and 10) Product Evaluation Challenge. These challenges are explored further in the following subsections.

4.6.2.1 Separation of Requirements from Requirements Change

The Request for Proposal documentation (as one of the sources of elicited requirements) did identify some high-level objectives, requirements and constraints for COTS selection. However, these were not sufficiently detailed to fully understand the needs of the users or to identify the finer-grained mismatches that lead to product customization.

Elaborations and analysis were needed on these incomplete and high level requirements to clarify and confirm the final specifications before the implementation could begin. Therefore, a substantial effort was dedicated during elicitation for uncovering and extracting changes from high level requirements to surface additional requirements.

Many of the high level requirement statements in the RFP carried the underlying potential of requirement changes. Because the origins of a customization (requirement) and a change coincided with each other, it was not possible for the analysts to easily distinguish the two. This separation was also challenging for the BAs because they were a) not domain experts b) had very limited product knowledge and c) had not been part of the contractual process.

4.6.2.2 Client Led Requirements Elicitation

In the CICP project the client was leading requirements elicitation process and the (initial) specification activities. The client led elicitation approach was carried out in two steps. In the first step the elicited requirements went through a *'local'* cycle of analysis, specification, verification, validation, prioritization and negotiation before they were shared with the vendor team. In the second step the client's project management team and their BAs collaborated with the vendor team to finalize the elicitation process.

FRNZ-AVA-06-2013 "[the client BAs] would invest a whole bunch of time developing requirements internally working with the business prior to a call with [the vendor]"

However back and forth communication and negotiation of changes between the local team and the globally distributed vendor consumed a lot of time and had a significant impact on the project schedule. Further repackaging and negotiation of the customization requirements placed additional demands on time and effort and contributed further delays to elicitation.

FRNZ-CKE-05-2013 "Time is certainly a problem … you lose twenty hours in a heartbeat…the biggest problem we have had is just the ongoing delays"

4.6.2.3 Lack of Product Knowledge for Feature Identification

Identification of the desired customizations by reviewing the existing product functionality was quite challenging for the client because of their limited product knowledge. The client BAs for the project were not part of the client's ICT staff and were contracted specifically for this project. The client's project management team expected that the contracted BAs would have sufficient knowledge and familiarity with the existing

complex COTS product to find matches and mismatches among the many workflows and features it had, as seen in the following statement:

FRNZ-MOL-05-2013 "[As a] business analyst, [I am] required to interact with the stakeholders, required to know about the out of the box system as a starting point and then to ask stakeholders for their requirements".

The understanding and establishment of the product feature baseline remained the biggest challenge for some of the BAs as suggested below.

FRNZ-MOL-05-2013 "The biggest challenge for me has been understanding the baseline product, so we have a piece of software it has got hundreds of things in it, biggest challenge is to know what those hundreds of things are."

Due to the complexity of the product and limited knowledge on the part of client BAs, identification and matching customization with the available functionality was challenging.

SRNZ-MOL-12-2013 "there are so many little questions, there are so many things where it is not 'obvious' on the screen. So someone has to know that and know to expose it and show it to us".

The client's uncertainty about the capabilities of the existing COTS product meant that the accuracy of feature match/mismatch remained variable. The client heavily relied on the vendor team members to help them in ascertaining the product features which caused further delays in elicitation due to physical separation of the teams.

4.6.2.4 Lack of Domain Knowledge for Product Internationalization

The client BAs were required to identify flag and specify USA-specific terminologies and features that were to be customized or translated into the regional (NZ-specific) context. This complexity was product specific which was due to the geographical, organizational, social and cultural differences between the vendor and the client work environments.

FRNZ-ISE-06-2013 "Most of their clients are in the United States and the research and ethics environment is 'kind of' different. There are some 'international' differences there. They have different modules but in American context… we want nothing of that"

However, product regionalization (US to NZ) and customization to fit the client specific needs became difficult because the client BA's had limited product and domain knowledge.

FRNZ-MOL-05-2013 "They [the stakeholders] have the subject knowledge I don't... there can be a piece of terminology on a screen and they look at it and go 'no, you cannot use that term here', and there's fifty reasons why, well I [BA] don't know that."

The vendor team showed their concerns regarding the lack of domain expertise in the client BAs when they visited the client site which was also acknowledge by the client team.

FRNZ-AVA-06-2013 "The whole point is that the analyst… assigned to this project is never a domain expert… I know more about the domain than they do"

As a consequence of some robust meetings within the client organization one of the client BAs resigned.

FRNZ-LEC-05-2013 "the vendors came here and saw our guys were inexperienced…so we did have to have a number of more robust team meetings, one of the BAs subsequently resigned,".

Domain expertise and product knowledge took some time to develop in the client organization.

4.6.2.5 Lack of Product Documentation

The client BAs repeatedly cited lack of documentation to be a problem for understanding the product, ascertaining the baseline, eliciting, communicating and even testing the requirements. Product functionality baseline had to be derived incrementally by the client BAs from using the COTS product itself.

FRNZ-CKE-05-2013"We work out what the base line is… well they have given us basically a two page instruction…. in terms of understanding in terms of what their existing system is … you basically work it out by testing, put different scenarios in and figure out… it is less than ideal"

SRNZ-MOL-12-2013 "As far as I can tell there is very scanty user documentation…Yes it has certainly been a problem to understand baseline".

The client and the vendor however, agreed to jointly develop the documentation for the product because there was very little available at the time of procurement. The documents which needed to be developed were: Integrated Development Environment Developer Manual, Technical Documentation, Administration manuals for users of the [the client] system[12]. The documentation work is still being carried out at the time of writing up of this study.

4.6.2.6 Lack of Availability of the Vendor's Expert Product Knowledge

The client considered that the expert product knowledge of the vendor representative was not always available when needed for clarifying or confirming product feature offerings.

[12] Documentation Plan 23 April 2012 Source Project Wiki

This was due to both the large time zone difference and significant demands on the vendor's time but it caused delays and frustration for the client team.

FRNZ-ISE-06-2013 "just timing, just the time of the day, four days of a week we can schedule a teleconference, which is 9:00 o clock in the morning that is 5:00 pm [their] time so that is a big constraint and among those times the [vendor team member] isn't always free.

The vendor however did not see the time zone or distance to be a barrier to exchanging knowledge and argued that technology lends itself well to abstract distance. The vendor instead highlighted staffing, management and people related issues to be the cause of problems, as stated below

FRNZ-AVA-06-2013 "I don't see that as much of an impediment. … I don't honestly buy into this distance argument at all"

From vendor's perspective there were *"regular meetings and frequent ad hoc calls"* with the client to transfer product knowledge. The vendor considered that there were *"regular and extended vacations of important [client's] project-related staff"*[13] *which was causing the problems for transferring required product knowledge*.

4.6.2.7 Conceptual Distance between Requirements and Product Features

Elicitation of requirements was challenging for the client BAs because some of the desired customizations were conceptually distant from the existing product features. The first were represented by user stories, workflow diagrams (Figure 4.11) while the second were embedded and often 'frozen' in the existing COTS product. The convergence and expression of desired customizations based on these conceptually distant ideas was a problem for the client team members.

[13] Follow up correspondence with Vendor 02 Oct 2014

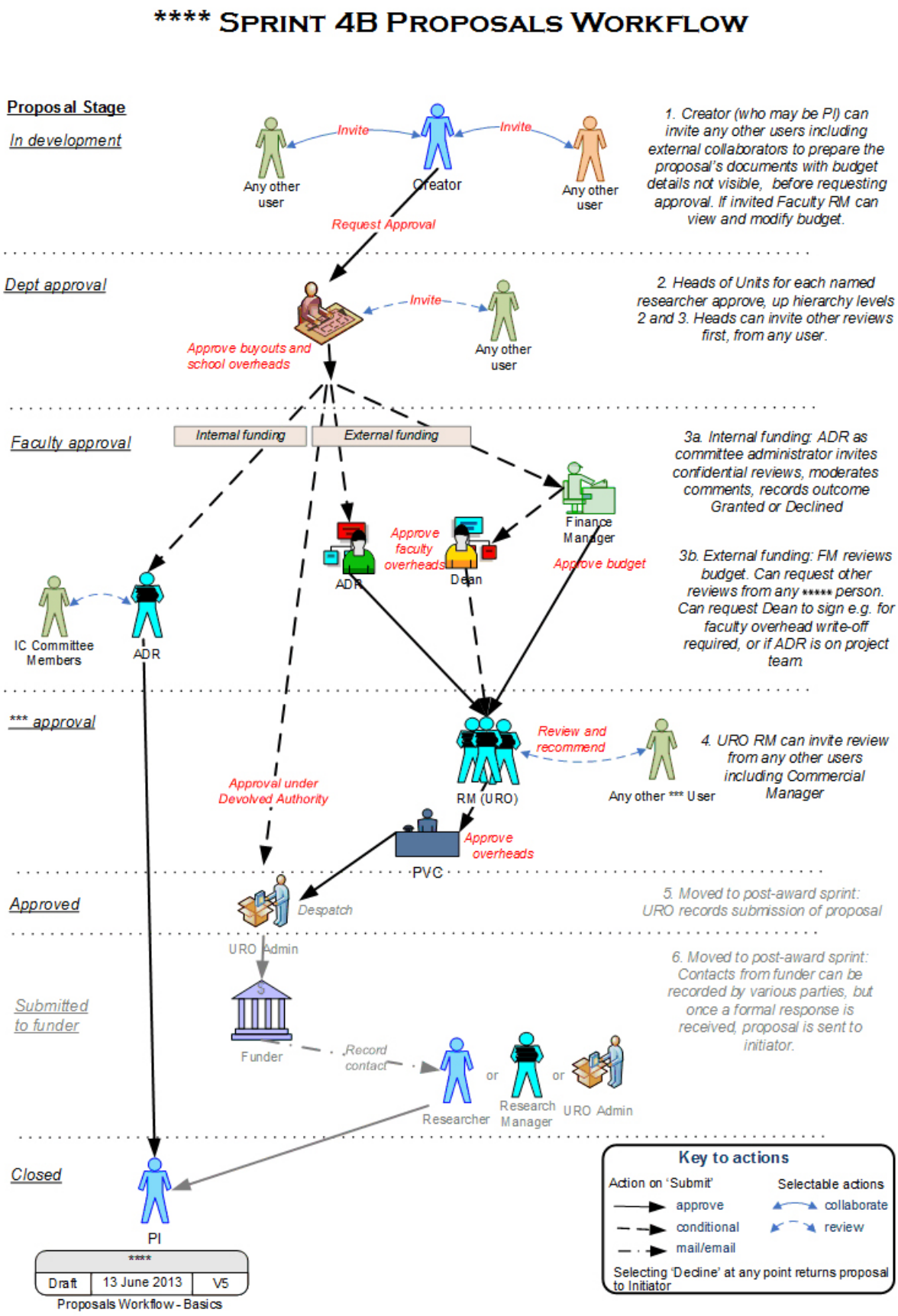

Figure 4. 11 Process Workflow Diagram for Customization in COTS Product

In some cases, the functionalities missing in the product no longer remained easily expressible in the client's desired workflow diagrams or user stories scenarios.

4.6.2.8 Product Evaluation Challenges

The client considered that limited opportunities to evaluate the product did not allow them to perform requirements elicitation adequately. The vendor argued that the product was well aligned with the client's business demands (otherwise it would have not been procured) and the client was provided several opportunities (demos, onsite visit, presentations, test access) to evaluate the product.

FRNZ-AVA-06-2013 "We had provided a lot of demos to people so we had prior presentations to the stakeholders. There were also onsite visits as well pre procurement…[The client] had four to six months of unfettered access to the system and as part of the pre procurement process."

However, the client reported that the sandbox (test) access to the product was not useful for product evaluation. The client team members, as suggested by one of the BAs, were not able to verify many features claimed by the vendor and were not able to understand the system without adequate supporting documentation.

FRNZ-ISE-06-2013"We have access to a couple of sandbox systems where they put their base product …there are features [that] don't work…we can't tell if that is because it hasn't been configured right or we don't understand … there is no documentation on it". It could be smoke and mirrors they say here is this feature you can use that and actually when you get down to it, it doesn't work. We have found a couple of those…they said that you could do versioning of budgets…. and [a BA] tried it out and it really 'did not work'."

Table 4.2 summarizes the challenges faced in the first lifecycle activity of requirements elicitation.

LCA 1 Requirements Elicitation -- Challenges	Challenge Description
Separation of Requirements from Requirements Change	Additional time, cost and communication overhead required to separate requirements from changes.
Client Led Requirements Elicitation Process	Additional time required to go through two cycles for requirements elicitation; one local and one global
Lack of Product Knowledge and Product Feature Identification	Additional time, cost and effort required to identify existing product features and matching them with customization requirements desired by the client
Product Internationalization and Lack of Domain Knowledge	Inadequate domain knowledge led to inability of the client to identify country specific domain mismatches (US vs. NZ context) and having to rely heavily on the vendor.
Lack of Product Documentation	Problems in identifying and verifying features offered by the product and testing the product without adequate documentation
Unavailability of Expert Product Knowledge	Unavailability of vendor's expert product knowledge when needed to clarify or confirm product feature offerings caused delays and frustration for the client.

Conceptual Distance between Requirements and Product Features	The desired system requirements and the product features were conceptually distant causing problems for elicitation and expression of desired customizations.
Product Evaluation Challenges	Limited opportunity to evaluate and verify product offerings even with a test access resulted in incomplete set of customization requirements.

Table 4. 2 Summary of Challenges in LAC1 – Requirements Elicitation

4.6.3 LCA2: Analysis and Negotiation Activities

The elicited requirements were initially recorded in many different formats such as user stories, annotated screenshots, workflow diagrams and Excel sheets. These elicited requirements were analysed and prioritized by the client BAs and were then negotiated for inclusion in the sprint scope by the product owner (in consultation with key client stakeholders). After the analysis of requirements from the client side, the product owner would send them to the vendor for his review and further scope negotiation.

The client and vendor team members would then collaborate to clarify and correct the incomplete, ambiguous requirements as well as negotiate requirements based on the contractual scope. As a result, the scope of the current sprint (the number of requirements or changes to be handled) would get decided. The vendor would check if the requirements or changes requested by the client fall within or outside the contracted scope in his opinion. Any expansions or reductions to the contracted scope would then be negotiated with the client. During these negotiations, some originally out of scope changes could be accommodated in the current sprint scope by the vendor without any cost to the client. Similarly, the client might defer or reduce (or altogether remove) the scope of requirements they perceive as contractually agreed from the project work. Such accommodations of requirements or changes by either party in this research are referred to as *'informal requirements change management'* activities. As a result of analysis and negotiations the two parties (and stakeholders) would agree on a high level set of agreed requirements for the product customization See Figure 4.12.

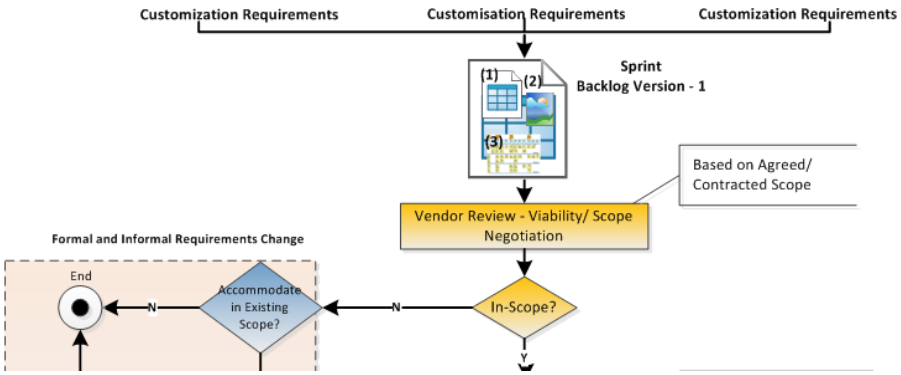

Figure 4. 12 Model in Practice-Analysis and Negotiation

4.6.4 CLCA2: Analysis and Negotiation - Challenges

There were several challenges faced by the team members on both client and vendor side while carrying out requirements/ change analysis and negotiation activities. These challenges are discussed here.

4.6.4.1 Requirements and Change Management Challenges

i. Conflicts Due to Informal Requirements Changes

Analysis and negotiation activities were fraught with challenges due to continuous changes in the requirements and the inability of both parties to agree on the 'contractual' scope. Separation of changes from requirements, i.e. deciding whether the proposed sprint requirements fall under the agreed scope or not, was very difficult for both parties. Often the client team considered that the desired customizations expressed for the upcoming sprint were within the contractually agreed scope however the vendor considered them as 'scope expansions'. This created friction between the two parties.

FRNZ-ISE-06-2013 "So until that point [scope signalling] we get the message 'that's all fine, no problem, we can do what you need' and now [that] he [the Vendor BA] has read the detail [he goes like] 'you guys are just dreaming, you have got no idea [of the agreed scope]".

The vendor considered that the product owner was '*jamming'* the scope by asking too much to be done in the sprint. The vendor pointed out that they were already making (informal) accommodations for the proposed expansion in the scope without charging the client for any extra billable time. This according to the vendor was a move to win over their first client in the New Zealand market.

FRNZ-AVA-06-2013 "Now partially as our first customer in NZ we were more liberal, I said I want you guys to get what you want, that doesn't mean open budget, that means we ate a lot of it [scope accommodation].

The vendor also suggested that their company suffered the consequences of accommodating '*over jammed'* scope of a sprint as stated below.

FRNZ-AVA-06-2013 "The scope is closely set within the sprint, we mutually agree to bite off a chunk of priority. We in good faith try to bite of as much, if not everything. In the history we tried to bite of everything for [the client] regardless of consequence to our company."

The vendor highlighted that fact that the impact of scope expansion (accommodating more than the agreed customizations/changes) was borne by the vendor company.

FRNZ-AVA-06-2013 "we could have easily just said, hey make that 10 days [sprint] to 20days, and of course then [the client's] budget would have gone through the window"

On the other side, the client considered that they were having to cut back many requirements which they thought was part of the agreed contractual scope. The client BAs were not happy about the extra effort and wastage of time on removing, restating and reprioritizing requirements after scope negotiations with the vendor.

FRNZ-ISE-06-2013 "as we find out the sprint is going to take 30 when we expected 10, then we go taking to it with a knife, and say 'maybe we can do without that' and seems that we are 'cutting back'…That's a hard trait, it is truly hard trait, we spend a lot of time 'defining what is needed'."

On the client side the informal accommodation and reduction of scope was very significant which made even the steering committee members very concerned.

FRNZ-IRW-06-2013 "from our perspective this [de-scoping] is our biggest concern... now that we are going to actually keep de-scoping [and] maybe in the end meeting the budget but what does that mean... the de-scoping would have some sort of major significance down the road."

ii. Unclear Requirements Baseline and Negotiation Challenges

The initial scope specified based on the perceived requirements baseline (vanilla product) in the contract had to be revised significantly. Several requirements had to be further evaluated, clarified, rewritten or modified. This gave rise to many changes in requirements and resulting in several challenges for requirements analysis. Furthermore, scope negotiations and management due to revised requirements as well as a moving baseline became big challenges for the client and vendor team. The client associated lack of understanding of the baseline due to product incompleteness as well as misrepresentation by the vendor about the product features.

FRNZ-ICK-05-2013 "as we looked more closely at the product we realized that it was in terms of its features and capabilities a lot less fully featured than [what] we have been lead to believe and that meant that we were essentially, rather than dealing with a packaged software, we were developing [one] from scratch, for certain areas of that product."

The vendor considered that the client did not stick to the product baseline and requested many changes in the baseline which caused problems with managing scope.

FRNZ-AVA-06-2013 "I always promoted a notion of baseline plus, which would be minor non material modification and baseline plus plus so that they will make things relative to a baseline so that they would take the least time and money and effort will be spend on modifications you know for [the client's] behalf that actually hasn't played out post procurement…pretty much across the board, [the client] abandoned the baseline in every single sprint"

From the vendor's perspective the cause of the frequent and customization intensive changes was the departure from the baseline and the client's desire to cater for their

stakeholders needs as much as possible. Both of which had implications for the client and the project.

FRNZ-AVA-06-2013 As [the client] got deeper and deeper into the application code and went deeper and deeper with its stakeholders, just decided that it wanted to make these things as perfect as possible and of course with some implications.

One of the client team members noted that deviations from the baseline and asking for significant customizations to the product resulted in monumental problems for negotiations and resolving issues.

FRNZ-LEC-05-2013 "if what we are proposing is very different from [the vendor's] base product the complications in not only the time involved but the error rate which has been very very high at the outset, ... the time involved in resolving those become a geometrical problem".

iii. Lack of Impact Analysis Information

Prioritization was done locally on the client side by the product owner and it had to be further negotiated with the vendor in synchronous meetings. In the absence of adequate product knowledge making informed decisions for prioritization and negotiation were not easy for the client team members. They did not receive adequate information about the change impact analysis e.g. which type of change has a strong impact on the system and hence requires more effort to be implemented. One of the BA's complained about the lack of adequate response regarding change impact and the effort or time required. He considered this information was critical to plan the scope for the coming sprint as suggested below:

FRNZ-ISE-06-2013 It is hard to get the signal from the vendor/developers as to what adds to the complexity and what is very easy? So we haven't really got good indication on that.

Change impact analysis was not carried out through a formal process. The client team members reported that the impact analysis was carried out informally on the spot (during client-vendor meetings) and the effort estimates were provided by the vendor without consulting his developers. The client team members were unhappy about the informal and inaccurate estimates from the vendor which resulted in delays.

The collaborative activities of analysis and negotiations became increasingly inefficient with the passage of time as the client could not rely on any of the impact analysis information or estimates. This was one of the major causes of frustration especially for the client team members and one of the major factors for the client-vendor relationship deterioration.

FRNZ-CKE-05-2013 "I would say that the vendor's estimates are not reliable, they say two weeks and end up taking two months"

4.6.4.2 Delays Due to Infrequent and Ineffective Communication

Another major challenge in requirements negotiations in this project was the extended time frame required for requirements review and negotiations. Infrequent and ineffective communications and the lack of key stakeholders' availability from both sides extended the required time for completing the analysis and negotiations.

FRNZ-ISE-06-2013 "we allocated two weeks for [specifications handover] which was meant to finish last Friday… we have succeeded in two meetings so far, because the vendor is unavailable , made himself unavailable or have some other plans".

The client considered that the vendor was responsible for delays whereas the vendor considered that the unavailability of client staff was the cause of delays.

"While the Vendor resources are noted as not always available (despite regular meetings and frequent ad hoc calls), regular and extended vacations of important [the client] project-related staff are not noted"[14]

The client BAs showed frustration over the infrequent communication and feedback during the requirements review as under

FRNZ-CKE-05-2013 "we have a got a project plan we expect them to get back and say these things are difficult and we should put them in a separate sprint … but you get nothing… I find that this just goes into the black box with no feedback, doesn't mean how many meetings you have doesn't mean how many emails you send."

4.6.4.3 Personality Issues and Lack of Product Owner Involvement in Communication

The client and vendor reported many communication difficulties and personality issues between the cross site team members during requirements review and negotiations. The product owner realized that it was being caused by the lack of involvement of the product owner's role in client vendor communication and tried to correct them through more involvement.

FRNZ-LEC-05-2013 "on the personal side, there have been a lot of communication difficulties, and personality issues, within the team and between the team and the vendor… I started to sit in on all the teleconferences the vendor was expressing frustrations and there was personal animosity between the two [the vendor and the PM] and [vendor] started demanding my presence"

[14] Follow up communication with Vendor 23 09 2014

The product owner considered that this change was also necessary to avoid downstream problems. This modified approach was not taken in the initial sprints which extended the time frame and made issue resolution a 'geometrical' problem.

FRNZ-LEC-05-2013 "[earlier] I used to stand back and let the BAs do their work, I have figured out that it was not actually enough because they [the BAs] go down wrong tracks very easily."

Table 4. 3 summarizes the challenges faced in the first lifecycle activity of requirements elicitation.

LCA2: Requirements Analysis and Negotiation-- Challenges	Challenge Description
Managing Requirements Change Informally	Separating requirements from changes and conflict regarding what was the agreed contractual scope. Informal reduction in the agreed scope by the client or accommodation of expanded scope by the vendor without charging the client. Bad relationship between the client and the vendor due to over-prescribed sprint scope negotiations and implementation which resulted in delays and low quality of code.
Frequent Requirements Changes	Frequent changes in requirements made scope management very difficult. Disagreements and figure pointing about not sticking to the product baseline (vanilla product) and departure from the agreed scope causing relationship breakdown on several occasions.
Formal Requirements Change Management	Using additional sprints as a formal mechanism to deal with frequent requirements change resulting in several project challenges. It in investment of additional time and effort and on several occasions pushed the final delivery beyond the planned deadline.
Lack of Impact Change Analysis Information	Unclear information about the change impact (complexity of development, time and cost) causing confusion and problems for negotiations. For the client prioritization and packaging of requirement for the sprint became a challenge in the absence of proper impact analysis information.
Infrequent Communication	Infrequent communications for requirements negotiations and the lack availability of the stakeholders from both sides (due to geographical and temporal distance) extended the required time for task completion
Lack of Product Owner Involvement and Communication Difficulties	Unavailability of the product owner in cross site communication caused conflict and friction between the two parties and resulted in a bad relationship.

Table 4. 3 Summary of Challenges in LAC2 -Requirements Analysis and Negotiation

4.6.5 LCA 3: Requirements Specification / Documentation

In CICP project the client team was tasked with specifying user requirements and changes. A combination of workflow diagrams, annotated screenshots and user-desired functionality (stored in spreadsheets) constituted the initial backlog for the upcoming sprint. Two or three iterations of communications were required before the client and the vendor could agree and finalize the detailed specifications and their priorities. These communications occurred through various asynchronous and synchronous channels.

The detailed specification of requirements took into consideration factors such as initial vendor feedback on the proposed sprint requirements, cost-benefit analysis, stakeholder needs and project schedule (See Figure 4.13).

Figure 4. 13 CICP Model in Practice- Requirements Documentation / Specification

4.6.6 LCA 3: Requirements Specification / Documentation Challenges

There were several challenges faced by both the client and the vendor team during requirements specification and documentation activities. The challenges faced included a) having to use a vendor supplied template for specification b) maintaining two systems for requirements and change specifications c) manually maintaining multiple versions of requirements specifications d) Conflicts over domain terminologies and e) Personality Conflicts.

4.6.6.1 Having to Use Vendor Specified Template

Following an agile practice (as prescribed by the project methodology) the client initially drafted stakeholder requirements in the form of user stories. The vendor did not agree to work with user stories and asked the client to use vendor-supplied spreadsheet templates to document requirements.

FRNZ-ISE-06-2013 "So we capture requirements, write up user stories and provide specification in the form that the vendor has asked for [spreadsheet templates]. He has said very clearly he does not want to hear user stories, he wants them [requirements] in spreadsheets and in a particular format"

Translating user stories into the vendor specified templates while maintaining user stories for their internal requirements management was considered a big challenge by the client team. Furthermore, the client also had to stop using their own (and preferred) spreadsheet templates to specify requirements. The following comments are taken from the project wiki concerning the adoption of vendor-specified spreadsheet template desired by the vendor.

Feb 06, 2013 [The Vendor] introduced a Workflow spreadsheet in an online meeting on 5 Feb 2013 and sent through the attached sample. This is obviously a format [the vendor] is comfortable with. He has put this material in front of a client, though I'd prefer to put process flow diagrams in front of clients and do the translation to the spreadsheet within

the project team. The format differs from the five-column table we adopted in Nov 2012. [15]

From the vendor's perspective the requirements specified in his spreadsheet format were more logically presented and could easily be transferred into the system. The vendor also suggested that the forced logic of this structure made it easier to communicate and negotiate requirements.

FRNZ-AVA-06-2013 "We [the vendor team] use our documents to communicate requirements …we ask the customer to document as much in a logical form and in a table driven kind of form using Excel as possible… We will look to the spreadsheets as a logically tabbed structure a self-referential structure to give us their requirements, then and only then can we get the business to be on a co-owner of those requirements"

4.6.6.2 Maintaining Two Systems for Managing Specification and Issues

The client team had to maintain (user stories and) specifications, changes in requirements and issues in *JIRA* as well as vendor specified spreadsheets. Due to the vendor's preference to not use *JIRA,* the client had to specify and communicate detailed requirements and changes from user stories into spreadsheets provided by the vendor outside *JIRA*.

FRNZ-ICK-05-2013 "we use the agile over layer in JIRA to track stories, but our actual specification is done outside of JIRA [in spreadsheets]"

The duplication of effort and creating and maintaining two separate sets of information for requirements change and issues was considered 'annoying' by the client team. Although the client team tried to persuade them the vendor team was not willing to use JIRA.

FRNZ-ICK-05-2013 "We have an issue tracking system that we use internally for logging defects and we try to persuade the vendor to use that as well but weren't very keen … they tend to use spreadsheets. It is a bit annoying because we first have to export the issues from our tracking system into a spreadsheet [which] they amend with their comments, that is separate from the issue tracking system so there is no record in the issue tracking system of their comments."

Furthermore, instead of putting issue status updates on JIRA, which the client thought was easy, the vendor team used spreadsheets. This created another detached pool of data for issue (and change) related information.

[15] Source Project Wiki: Business Analysis and Requirements Section

FRNZ-ICK-05-2013 "in terms of his updating that issue which is common and simple, they use spreadsheet rather than putting them in JIRA".

4.6.6.3 Manually Maintaining Multiple Versions of Requirements Specifications

The client was managing user stories and specifications both in JIRA and the project wiki. Therefore, a significant effort was required not only to convert the user stories and issues into the spreadsheet templates but also to manage their multiple versions shared across sites.

Analysis of the project Wiki revealed that several versions of requirements specifications had to be maintained by the client (See Figure 4.14). Some of these sprint spreadsheets had up to six versions and with multiple specification sheets embedded in each version.

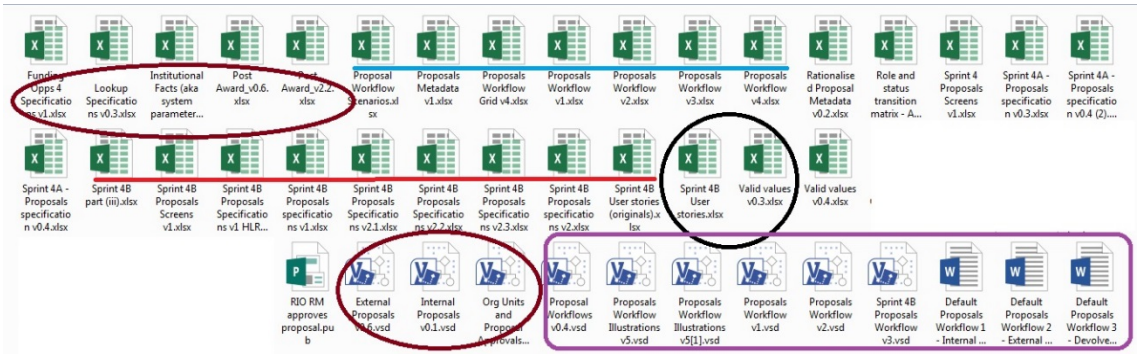

Figure 4. 14 Versions of Requirements Specification and Related Artefacts

In addition, the client team had to maintain and keep available other requirements related documents (such as images, notes, comments and requirements related communications) in the project wiki and bug tracking system.

The vendor however, was of the view that there were not many versions of spreadsheets to be maintained so it was not considered a major issue.

FRNZ-AVA-06-2013 "we [the vendor team] use our [spreadsheet] documents to communicate requirements, but they tend not to be as many versions of that as you might anticipate"

4.6.6.4 Disconnect Between JIRA and Sprint Specification Versions

There was a disconnect between *JIRA* and various sprint specifications. The client team members struggled to match and map client-vendor team communications corresponding to individual issues recorded in *JIRA* as seen in Figure 4.15. Issue related artefacts were sometimes not attached with the issue on *JIRA* which did not allow the client team members to view all information related to an issue at one time. This was again considered

quite challenging for the client team as it almost doubled the document maintenance and information retrieval effort.

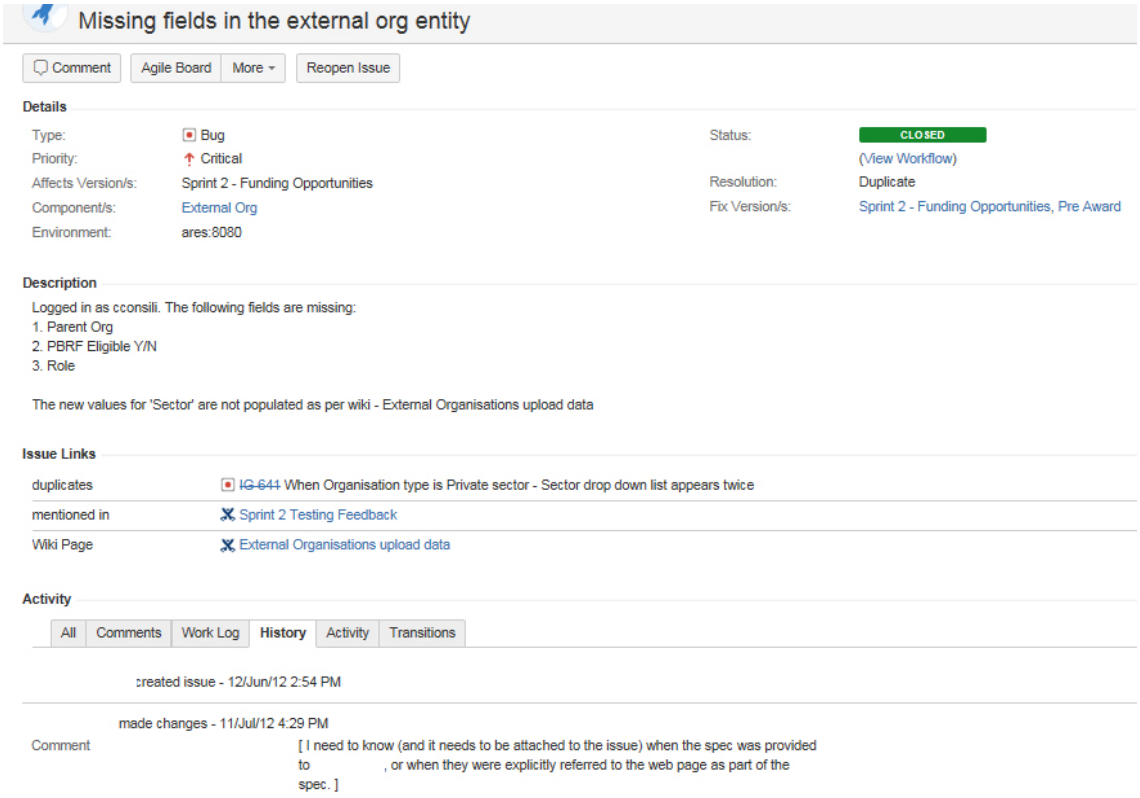

Figure 4. 15 Showing Disconnect Between JIRA and Sprint Specification Versions

The project wiki maintained by the client team remained useful only to the client team internally as in most cases the vendor team (including developers) did not use it at all. Table 4. 4 summarizes the challenges faced during requirements specification activities.

LCA3: Requirements Specification Challenges	Challenge Description
Having to Use Vendor Specified Template	The client had to use vendor specified templates for writing specifications rather than working with user stories or spreadsheet templates designed by the client BAs. This was considered frustrating by the client team members.
Having to Maintain Two Systems for Managing Specifications	Maintaining specifications by means of user stories and issues on JIRA for the client's in-house use as well as on vendor provided spreadsheet templates (for specification) was considered very challenging by the client team members.
Manually Maintaining Multiple Versions of Requirements Specifications	The client did not use any requirements management tool and because the requirements and specifications were scattered in many spreadsheets with multiple versions. Maintaining multiple versions of these artefacts was very challenging for the client.
Disconnect Between JIRA and Sprint Specification	The client was using JIRA for recording user stories and tracking issues while the vendor worked outside JIRA for both these activities. This created a disconnect between the two systems maintained.

Table 4. 4 Challenges Faced during Requirements Specification Activities

4.6.7 LCA4: Requirements Validation

The client and vendor teams went through requirements validation cycles taking two to three collaborative sessions prior to finalizing sprint requirements. These iterative sessions were called *'specifications handover'* within the project. A duration of two weeks was assigned for the handover completion. These sessions involved a combination of requirements related activities such as analysis, verification, clarifications, negotiation and checking for requirements consistency and completeness. During the handover detailed customizations were iteratively discussed, analyzed, validated and even negotiated to establish shared understanding and finalize the specification.

These synchronous *specifications handover* sessions were central to finalizing customization requirements and change details before implementation. S*pecifications handover* sessions were carried out from remote client and vendor locations in the USA utilizing web conferencing tools, phone calls, and shared electronic artefacts.

FRNZ-ISE-06-2013 "once we [the client] decide what is in the sprint, we have got the [specifications] handover process…we have allocated two weeks period to hand over [specifications] so that we are confident that the requirements have been understood by the vendor. In that period he [the vendor] is meant to assimilate that, ask questions, and clarify things so that his understanding is good enough that he can hand it over to the developers."

The resultant artefact from this phase is the final version of sprint backlog agreed by the vendor and the client team for implementation as shown in Figure 4.16

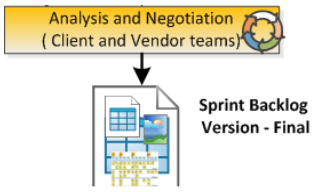

Figure 4. 16 Requirements Validation- CICP Model in Practice

4.6.8 LCA 4: Requirements Validation- Challenges

4.6.8.1 Personality and Role Conflicts

Both the client and the vendor team members were having problems in finalizing specifications which required both parties to collaborate over requirements. The client and vendor BAs were having personal conflicts and blaming each other for things like the inability to share roles and lack of expertise. These problems led to two of the client BAs resigning from their jobs.

The vendor also suggested that the client BAs were not doing their job properly when specifying requirements and were leaving requirements ambiguous. He considered that one of the client BAs who could easily get agitated was not suitable for the role he was playing.

FRNZ-AVA-06-2013 "I thought [the BA] was doing pretty well, I think [the BA] is just a kind of personality that gets flustered very easily, a flustered person should have no role in requirements development or project management…. It takes a certain kind of person."

The client team members however considered that there was a problem dealing with a particular role on the vendor side who they thought was not happy in sharing his role with the client BAs.

FRNZ-ISE-06-2013 "[the Vendor BA] is not used to sharing that role … we are right at his face, doing this work on behalf of [the client] … we have had a lot of friction with him over that."

The client BAs knew and reported that the vendor BA wasn't happy working with them and making up things that would cause conflict.

FRNZ-ISE-06-2013 "So effectively, either the specs are badly done or not done well enough or not done how he likes them".

It was suggested by the client BAs that it was a personality problem. The fact that the vendor BA did not like someone else is taking his place or role on the client side was causing many conflicts.

FRNZ-ISE-06-2013 "My picture is, he is this big business analyst and we are acting at his place and I don't think he likes that. He is pretty sensitive to criticism. If he points out something missing in my specs and I will fix it up and send it back and I will bear it. But the not the other way around no he gets quite prickly and I think he will occupy a lot of telephone time about that. So it is much more a personality issue".

4.6.8.2 Terminology Differences

Although the vendor reported that the domain terminologies did not really pose a problem other than label changes and the specifiers but in reality there were many arguments regarding terminology differences. It led to many conflicts between the client BAs and the vendor and even led to personal clashes where team members had to intervene. Consider the following detailed quotation from the vendor regarding problems discussing and agreeing on domain terminologies.

FRNZ-AVA-06-2013 "when I have to describe the difference between work flow and a business process as a means to manage to advance the state of a record, which is universal definition and a collaboration being a soft coordination between the parties within a stake. … I had to explain to [the BA] that it is

really important that you use these terms precisely, because if you call it workflow we will interpret it in the universal definition of this term. If you term it as collaboration it means something else. Something as trivial as that would set [the BA] off, as if I was trying to be 'semantic' with [the BA].

"they would describe a workflow and I would say you are calling that 'workflow' but let me make sure I understand you. When you say these people are working together is anyone changing the state of the record, NO, it is all within this agents status. I would appreciate then, in the document that you send us, do not use the word workflow do not use the word process there, call that collaboration because collaboration is joint work without advancing a state. And something, honestly, as objective as that I would get a defensive reaction. So and at that point that becomes the end of conversation. I don't, I can't further the dialogue when I am working with a sub power resource."

One of the client BAs suggested that they were doing their job and it was a difference in perceptions that it was seen as adversarial.

FRNZ-ISE-06-2013 "From [the vendor's] point of view we are changing all the time, from our point of view we are clarifying and providing things in the form that he needs. So it is interesting the difference in perception".

Continuous conflicts over domain terminologies, personalities while discussing specifications resulted in bad relationship between the client and vendor throughout the project and also resulted in resignation of two client BAs.

4.6.8.3 Delays Due to Coordination Issues

Synchronous communications were central to the successful analysis and negotiation of the customization requirements. Although such synchronous meetings were highly desirable, they proved to be difficult to organize because of the geographical distance and difference in time zones. Even during overlapping hours it was difficult to align the availabilities of team members from the client and vendor side. For example, in one instance the vendor was forced to participate in the meeting from a public place in order to avoid cancelling it.

Time zone difference, few overlapping hours, lack of availability of resources, and the difficulty of arranging synchronous meetings all contributed to delay in the requirements related activities which is evident from the following statement.

FRNZ-ISE-06-2013 "this is the only one [sprint] I have been involved in directly. I have seen a few others come by in the other role. They have tended to go long."

The challenges related to requirements validation are summarized in Table 4.5.

LCA4: Requirements Validation Challenges	Challenge Description
Personality and Role Conflicts	Personality issues, similar role conflicts (client & vendor BAs) caused delays and bad relationships during requirements validation activities.
Domain Terminology Differences	Continuous friction over domain terminologies while discussion specifications actually resulted in major conflicts the two parties and resulted in resignation of two of the client BAs
Delays Due to Coordination Issues	Coordination issues due to geographical and temporal differences, "Specifications Handover' sessions were carried out by collaborating over requirements specification using

Table 4. 5 Summary of Requirements Validation Challenges

4.6.9 LCA5: Implement-Test and Fix Activities

The vendor was responsible for implementing the agreed on specification in a period between two to four weeks depending on the stage of the project and the scope of the sprint.

The developers carried out their testing however from the customer's perspective the quality assurance and testing were carried out at the completion of the development iterations by the client. The client team tested the release and reported the bugs back to the developers for fixing them before the final user acceptance (See Figure 4.17).

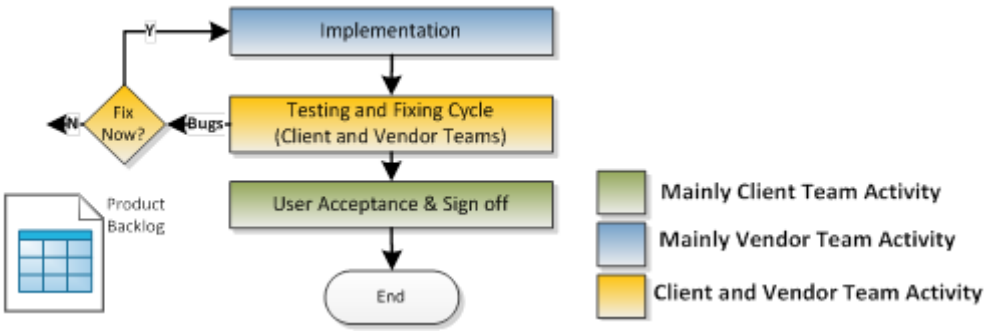

Figure 4. 17 Implement-Test and Fix Phase

4.6.10 LCA5: Implement-Test and Fix Phase Challenges

The stakeholders experienced some challenges during implementation and testing activities as well. The main issues from the client's perspective were long delays between releases and the lack of quality in the developed code. From the vendor's perspective the challenges resulting from under specification of requirements and the related ambiguities that led to problems in implementation. These challenges are discussed here.

i. Software Intensive Changes Implementation and Delay

According to the client team members there was no formal change request process for changes in requirements. Changes were treated new requirements and the client and

vendor specified one area of functionality they wanted to cover in a particular sprint (based on the contractual or agreed scope). Once that was decided implementation started.

FRNZ-ISE-06-2013 There is no change request process, no. I guess this is the feature of the 'agile' process…there is no overhead in saying we need a request for change. Change is assessed like any other new feature."

The project went through many software intensive changes. Some of those changes had to be handled in additional sprints (other than those initially agreed sprints prescribed in the contract). This obviously had implication for not only analyzing requirements and negotiation process but also for the project budget and time line. The vendor described the addition of such non-prescribed (additional) sprints as the formal requirements change management process.

FRNZ-AVA-06-2013 "change management is not within the sprints it is between the sprints. So change management is now when [the client] says, we don't want to move on to the next officially prescribed sprint, the business users have asked us for a bigger work area and want more work to be done and we are going to insert a new sprint. And that would be a change management process"

The team members were burdened with the additional collaboration, coordination and communication overhead due to the analysis and negotiations of changes in requirements. Changes were first analyzed and negotiated internally at the client site and then with the client before both sides could agree on the finalized changes that could go further for detailed specification.

4.6.10.1 Delay in Releases

From the client's perspective the implementation took much longer than the time specified in the project plan. Furthermore, during the implementation cycle there was not much feedback from the vendor side which left the client with little awareness of work status. For the client the long delay between releases was very frustrating not just because it was costing them money but also because client team members were sitting idle waiting for the release so that they could start working on it.

FRNZ-ISE-06-2013 "a burning issue [is] you can't rely on getting a sprint through that's going to get resolved according to your project plan. You've got to reschedule the whole thing and this is costing money and time, we have got people sitting on the bench waiting for the software to get developed, somebody is taking longer than expected"

Similarly, other BAs from the client team shared their experience of having to deal with release delays

FRNZ-CKE-05-2013 "We have got a project plan I expect them to get back [about requirements] and say, the functionality is difficult , we should put it in a separate sprint… but you get nothing , all you get back is , 'this is easy , oh yes I completely understand your requirements … and a month later, you are waiting on [a release]"

The client team members similarly complained about the bug fixing duration once the release was tested. The client BAs suggested that there were undue delays from the vendor side to get the problems fixed and it was more frustrating because there was no feedback from the vendor team regarding the issues.

FRNZ-CKE-05-2013 "we for instance, recently sent them all our defect but we are still waiting weeks and weeks and weeks for them to fix anything…this just goes into a black box with no feedback"

4.6.10.2 Low Issue Resolution Rate

The issue resolution rate was not considered good and the quality of resolved issues was low according to the client team members. This was considered challenging because issues required several attempts before getting fixed and the relationship between developers and the client BAs continued to suffer.

FRNZ-ISE-06-2013 "[the developers say] 'out of the thirty issues you [client BAs and Testers] gave us, here are the fixes, we have put them onto your environment- retest'. And amongst those the success rate is about 50%. We tick off half of them we find new ones; things broken because of that and we find some of them are simply not fixed or not fixed completely. Generally the 'severity' goes down, and some of them are not fixed at all".

The client complained to the vendor for low work quality and delays in resolving issues. The client attributed these problems to incomplete and rushed testing, release planning and other upstream problems. The also pointed out to the absence of any dedicated testing team member on the vendor side so the bulk of the testing has to be performed by the client team members (BAs and the tester).

FRNZ-ISE-06-2013 "their approach to testing is not very mature. There are no testers [at the vendor site], we do testing and there is someone [a tester] who is permanently in the testing role.

The vendor on the other hand considered that the client's over-ambitious scoping, lack of scope management and inadequate representation of the requirements as the cause of low quality of work.

FRNZ-AVA-06-2013 "…you cannot under specify and expect explicit development of the things that were unstated. Also you can't give a big ambitious scope and expect it all to be done at the same level of quality"

Furthermore, the vendor team was of the opinion that due to the accommodation of new requirements the quality of the work suffered, however they fixed the problems in the developed code without charging the client.

FRNZ-AVA-06-2013 " you find that there are some bugs and of course there are some bugs, that's because we did everything with almost no testing but then we fixed it without any billable time"

4.6.10.3 Inappropriate Communication of Issues and Delay

Both the client and vendor teams faced delay due to inefficient issue communication process on the client side. The client used to send the identified bugs (or suggested changes) to the vendor without priority and severity categorization. In the absence of any priority or severity, the developers were not sure which issues to resolve first so they applied a sequential (but inefficient) approach to fix issues shared with them.

FRNZ-LEC-05-2013 "They [the vendor BA and the team] were sort of overwhelmed by them [the defects], that makes them grumpy, irritable, they were fixing them from number one to number sixty which was an ineffective way"

Realizing their mistake, the client team members considered changing their practice of how they communicated the bugs.

FRNZ-CKE-05-2013 "recently we sent them all our defects but we are still waiting weeks and weeks and weeks for them to fix anything. What we should have done is just send them the severity one defects"

It is clear that quality of code issues were causing problems between the client and the vendor. These challenges are summarized in Table 4.6.

LCA6: Implement-Test and Fix Phase Challenges	Challenge Description
Software Intensive Changes Implementation and Delay	Software intensive change implementation contributed to delays in several ways. These challenges contributed to the overall development time, communication and collaboration overhead and delay in release cycles and prevented fast issue resolutions.
Delays in Release	The delays in releases caused the project to get derailed from the plan as per the client and caused frustration among the team members who were expecting regular and more frequent release.
High Volume of Defects and Low Issue Resolution Rate	High volume of defect and low issue resolution rate again was a problem faced by the client team members which they thought was due to poor testing methodologies and poor process on the vendor side. The vendor however, thought that it was due to improper specifications and over jamming the scope for which they got very little time to test.
Communication of Issues or Bugs	The mechanism initially adopted by the client team to send out issues that needed fixing without priority assigned to them was a problem causing delays in response. The approach was later changed by the client.

Table 4. 6 Implement-Test and Fix Phase Challenges

4.6.11 Requirements Traceability as a Supporting Activity

Data on traceability activities was mainly available from the client side. The client was using (*JIRA*) as their issue management system. JIRA was also used to support traceability information for the CICP project. Requirements and changes were stored in the form of epics (user stories containing high level requirements), user stories, new features, tasks and issues. Each of these items was identified by a unique identification number. Epics had bigger scope and were broken down into some smaller scope user stories, each with a link backward (to the originating epic) and forward to the release versions.

4.6.12 Requirements Traceability Challenges

The only challenge related to requirements traceability reported by the client was the additional effort required to maintain traceability information both in spreadsheets as well as in JIRA. Since the vendor team, did not like working with the information available in *JIRA* therefore the client had to convert the user stories, issues and tasks into vendor specified spreadsheet templates. This required additional effort by the client team to drive almost two mechanisms for traceability of requirements and therefore was reported as a challenge.

FRNZ-ICK-05-2013 "I don't think they have got a tracking system, I think they tend to use spreadsheets ...It is a bit annoying because we first have to export the issues from our tracking system into a spreadsheet then of course they amend the spreadsheet with their comments, that is separate from the issue tracking system so there is no record in the issue tracking system of their comments."

The analysis of the information about user stories, epics and issues available on JIRA revealed that

- Traceability information available on the client issue tracking software was not fully maintained and was mainly used by the client team.
- There was no trace backwards from the epic to the contract or request for proposal. However, epics and user stories were traceable forward (to the individual sprints in which they are implemented) and backward (to the originating epics or user stories).
- A total of 2294 issues were reported on JIRA (as of 31/12/2014). Out of these 994 did not have a forward traceability link to the version they were going to impact.

- There was a combination of 717 issue and stories which did not specify forward traceability as to which component they will affect. Similarly, there were 1041 issues which did not specify forward link to the version they were fixed in.
- The test scenarios were traceable to the related components in the system but were not linked to the corresponding user stories epics, issues or the design documents.
- 407 issues and/or bugs did not have any information about which version or component they would affect and in which version they would get fixed.
- The vendor placed release notes to inform the client about the implemented epics, user stories or tasks in a particular release. However, there was no information available on the vendor's intra-organizational traceability activities.
- Data and artefact analysis did not reveal the use of any traceability matrix by either the client or the vendor teams.

Traceability is an important supporting practice in agile development (Espinoza & Garbajosa, 2011; Paetsch, Eberlein, & Maurer, 2003) however, investing in detailed requirements documentation including traceability is not considered a cost effective approach (Paetsch et al., 2003). This could be seen as one of the reasons why artefacts such as requirements traceability matrix was not used in the CICP project.

4.7 GSD Related Challenges for the CICP Project

Challenges related to the global distribution of the client and vendor teams were ascertained from the interview data and participant experiences of the CICP project. Interviewees were first asked to first talk about the general challenges faced during the project and then share specific challenges relating to the three distance dimensions a) geographical distance b) temporal distance c) distance due to differing languages, organizational and country cultures. The perceived GSD challenges were then inductively explored, analyzed and categorized using *Thematic Content Analysis Technique* discussed in Chapter 3 Section 3.5.2.

The challenges introduced due to distance exacerbated problems faced during the lifecycle activities of requirements changes. These challenges are presented here as barriers to collaboration in a similar manner as discussed by (Noll and Beecham 2010).

4.7.1 Collaboration and Coordination Barriers Introduced by Distance

Geographical distribution of the project meant that the client was dealing remotely with two segments of the vendor organization; the coordinators and the project development

team. Similarly, the vendor was working with two remote stakeholder groups, the project management and the users.

4.7.1.1 Distance and Collaboration Challenges

Physical distance proved to be a major challenge for requirements communication and work collaboration especially because the opportunities for informal and face to face discussions were very limited. The management team from the client suggested that activities such as communication of requirements, change understanding and baselining all were challenging due to distance. One of the BAs stated:

FRNZ-MOL-05-2013 "Generally communication is harder, to get to work [collaboratively] is harder. It is much easier to walk up to somebody and have a discussion rather than having to organize a teleconference"

4.7.1.2 Distance and Challenges to Understand Requirements Baseline

The client team also considered that the distance and the inability to talk to the vendor team face to face was creating problems with respect to understanding the product and its baseline. This was also creating problems in understanding the impact of changes requested. The developers were missing on the contextual information required to understand and implement requirements was also missing.

FRNZ-MOL-05-2013 "when you are X thousand miles apart, you do your best, but you are not going to get that direct feedback from your stakeholders.

4.7.1.3 Distance Hindering Visibility

Geographical distance and remote development introduced lack of visibility and lack of personal accountability as stated by one of the client BAs. This causes the development team to either miss out on the requirements understanding that could have been possible through more background information which was needed for better decision making. One of the BAs suggested that when a direct link with the customers is broken then it becomes difficult for the developers to feel any personal accountability for the user requirements they implement.

FRNZ-MOL-05-2013 "developers miss out on the visibility so they miss out on spotting some of the requirements that weren't in the spec or some of that back ground information that will allow them to make those little detail design decisions heading in a better direction" accountability is lost when it is remote development not necessarily just global but certainly in the global.

4.7.1.4 Distance Hindering Issue Resolution

Due to the physical distance there were few opportunities for face to face communication between the client and the vendor team. Project related issues were not getting resolved over the phone and the tension between the two parties (client and the vendor) became very high.

FRNZ-LEC-05-2013 "They [the vendor BA and Lead Developer] were here for a week and they got to know us. That was absolutely necessary because things were getting very very tense on the telephone"

4.7.1.5 Distance Hindering Good Client-Vendor Relationship

Distance and the inability to have face to face communication was one of the main causes of relationship breakdown which happened on several occasions. The project steering committee had to start rebuilding the relationship but it continued to suffer from distance related challenges.

FRNZ-IZN-07-2013 "The relationship with the vendor has broken down well it has broken down on several occasions. Broke down very badly towards the end of last year and [members] on the steering committee have tried to do as much damage control and relationship building and support the project team as much as possible. But there are still some [problems]"

In order to compensate the distance factor the client team arranged onsite visits. It was realized by both parties that the onsite visits were necessary for resolving issues.

FRNZ-ISE-06-2013The vendor has been out here two or three times, that has been needed, that really does get things moving, [the vendor]came up last time with his senior developer, and in a week they worked through a fifty or a hundred bug fixes and they were tested on the spot. If we could do that, it would be good but that is a bit too expensive.

The client considered the visits were crucial in speeding up bug resolution and understanding the product so there was another planned visit by the vendor team. However, that did not happen till the end of the project.

4.7.2 Barriers Introduced by Temporal Distance

Temporal distance, according to the client, was the major issue causing problems for all collaboration activities the CICP project.

4.7.2.1 Temporal Distance Causing Coordination Issues

A large time difference between the remote client and team members (between 16 to 18 hours) resulted in very few overlapping hours which was reported by the client team as the main challenge for cross site communication.

FRNZ-CKE-05-2013 "When I get to work I have got a couple of hours, if he does release something and there is some major issue with it he has got only two hours to fix it. Otherwise I have lost another whole day."

Temporal distance caused collaboration and coordination challenges which were not eliminated even with frequent conference calls. Coordinating conference calls was not easy as either the parties could not (virtually) meet or could not effectively coordinate work due to the lack of overlapping hours It was difficult to arrange teleconference in case an issue needed to be resolved immediately as suggested here:

FRNZ-ISE-06-2013 "Just timing, just the time of the day, four days of a week we can schedule a teleconference, which is 9:00 o clock in the morning that is 5:00 pm New York time. So that is a big constraint. And among those times [the vendor BA] isn't always free.

FRNZ-MOL-05-2013 "Time zone, is a classic problem you need phone call or video conferencing as a proxy for face to face….[but] 3 o' clock in the afternoon their time is 7:00am in the morning here…Mondays are out because it is Sunday there. If you missed that 7:00 am or 8:00 am phone call it is end of their day, it is a days delay before the next one [synchronous meeting] can be scheduled, so when there is an issue it is a day's delay."

In the client's opinion if the development sites were located onshore or near shore, project communication with the vendor team would not have been as difficult.

FRNZ-ICK-05-2013"if they were in Wellington vs. in the north Hemisphere you would probably pick up the phone and catch them at 4 o' clock so I mean that makes it harder"

4.7.2.2 Temporal Distance Hindering Feedback and Issue Resolution

Another important challenge raised by one of the client BAs was the absence of a direct feedback loop due to both temporal and geographical distance.

FRNZ-MOL-05-2013 "so the time zone is really the additional problem certainly when you are outsourcing particularly globally… you lose the direct feedback loop between stakeholder and the developer".

Temporal distance and lack of overlapping hours contributed to reducing the number of opportunities for the client to seek feedback from the vendor. One of the client team members suggested that this happened not only because of the different in working hours but also because of missing out working days.

FRNZ-ICK-05-2013 "part of their weekend occurs during our working week so there would be time when they are not available during your working time to answer your question".

He considered that in collocated development where the interaction between the stakeholders and the development team is frequent there would be an added incentive to resolve issues.

FRNZ-MOL-05-2013 "where a particular individual might have been responsible for that module and he knew if he got it wrong the shipping lady upstairs would come down and talk to him and say 'you have got it wrong', so there is incredible incentive for a developer to really understand what people need and go that extra mile".

From the vendor's perspective temporal and geographical distance were not a barrier to the project activities

FRNZ-AVA-06-2013 "So I don't think these barriers have virtually anything to do with it [the problems]…we communicate and collaborate very well through Skype … we also likewise use GoTo meeting…distance has been abstracted successfully."

4.7.3 Cultural and Language Barriers to Collaboration

On the question of language and culture differences there were mixed replies both from the client and the vendor sides. Some interviewees believed cultural differences did introduce challenges while others did not.

4.7.3.1 Country Culture Undesirable Dominance

The client team members considered that the vendor was very vocal and dominating which was due to their culture (as American). It was in contrast to the NZ culture where people tend to be more polite and nice and as a result they were at time not taken too seriously.

FRNZ-ISE-06-2013 "American culture, it just seems that unless you are talking a 100% of the time people think you are a looser and just not to be listened to".

Consider the following statements where one of the steering committee members considered that cultural politeness from the client was one of the reasons of unfavorable outcomes of the project for the client:

FRNZ-IZN-07-2013 "Australia is far more like US and NZ is more like UK. So if you had a British company having to deal with NZ university you would probably get a better result on cultural basis… But this 'up in New York' which I feel [is] fairly aggressive [vendor] trying to deal with a [client organization] in NZ… [the client team] people are very polite and gentle and soft spoken and 'collaborative' and you know they are 'nice'".

There were cultural differences in how one side perceived the other and how one side could dominate the other.

FRNZ-IZN-07-2013 "the culture of NZ is quite different… even from Australia … radically different and people they think they are Superman Smarts… and they are not. America I guess has always seem to me to be far more like Australian.

The language was similar for both the client and the vendor teams but it was not the same according to one of the steering committee member which was seen as one of the reasons for the challenges in the project.

FRNZ-IZN-07-2013 "Two nations separated by common language really."

However, from the vendor side the differences between language or culture did not play a significant role in any challenge faced in the project

FRNZ-AVA-06-2013 "I don't honestly buy into this distance argument at all. Honestly I would think that it is looking for problems where they don't exist… people want to find fault in everything but where the fault lies."

4.8 Role of Collaborative Technologies (CTs)

This section deals with the second research question of this study that relates to the role of collaborative technologies in managing requirements change activities. To answer this question, the interviewees were asked to provide their perceptions on the overall use and role of collaboration technologies (CTs) in carrying out requirements change related activities. The participant experiences and perceptions of the role of CTs in the CICP project were then analyzed using Thematic Content Analysis (Chapter 3 Section 3.5.2).

Figure 4.18 provided a list of the collaborative technologies, their purpose and frequency of use in the CICP project for both within site and cross site collaboration.

Collaborative Tool	Within NZ	Purpose of Use	Between Client & Vendor	Purpose of Use
	Frequency		**Frequency**	
Email	Very often	Specs, discussion documents, questions which don't need immediate response	Very Often	Clarification, specifications, understanding, negotiation
Phone	Often	Clarification, Understanding	Never	
Audio Conference with Screen sharing (GoTo Meeting/Skype)	Hardly	Project progress meetings, if someone has to be offsite	Often	Goto Meeting used frequently for demonstration of baseline, understanding, clarification of specifications
Electronic Artefacts	Very often	Specs, Test Plans, Issues yet to be registered in Jira	Often	Numerous specification documents and discussion documents
Project Wiki (Confluence)	Very Often	Stories, Issues, Documents	Often	Storage, sharing and Collaboration
Bug Reporting Tool (JIRA)	Very Often	Issues	Often	Storage, sharing and Collaboration

Figure 4. 18 The Purpose and Frequency Use for Different Collaboration Technologies

In the following sections the role of collaborative technologies (both helpful and hindering) is discussed in detail.

4.8.1 Helpful Role of JIRA and Confluence

The client project team used a combination of JIRA (a project and issue tracking software) and Confluence (a team collaboration software) for their collaboration needs. JIRA (with the *Greenhopper* overlay) allowed the project team members to create, store, track and manage requirements and changes (in form of epics, user stories, improvements, new features and issues) and test related items.

FRNZ-CKE-05-2013 "We have got all our stories logged in JIRA, we have got our entire defect logged".

Confluence was used by the client team as a central space for client team collaboration for project and requirements related artefacts to collaborate. The vendor team did not actively use Confluence for collaboration activities. The client team however relied heavily on Confluence for team collaboration and to maintain and update many versions of various project artefact.

FRNZ-ISE-06-2013 "We have a wiki which has issues and users stories on it and also the technical documentation"

Project artefacts were maintained in Confluence under specific categories such as16

- Project Background, Analysis, Scope, Vision and Design documents
- Project Plans Communication plans Release Management and Change Control
- Project Testing Approaches and Acceptance Criteria
- Project Issues, Risks Security
- Sprint Specifications & Deliverables
- Onsite Meetings and Teleconferences
- Meetings On-Site Sessions Teleconferences
- Vendor Information
- End User Documentation

In addition to user stories, detailed requirements and change specifications were written and shared in other formats such as, Excel sheet templates (provided by the vendor), workflow process diagrams and GUI customizations. Client and vendor teams shared and

[16] Project Wiki (*Confluence*) File Lists

collaborated over the actual specifications (written in spreadsheet templates) outside *JIRA* and *Confluence*.

These additional formats were stored in *Confluence* and made accessible to both the client and vendor teams. The client team used these tools (*JIRA* and *Confluence*) to set up a central space with directory-like structure that could be used to store artefacts for collaboration

FRNZ-ISE-06-2013 "We have a shared space that this is where you keep communicating documents where you keep meeting minutes, issues registers and the rest of it."

The BA considered Wiki to be an important and better tool for collaboration than using shared folders because it allowed an audit trail of documents shared by the team members.

FRNZ-ISE-06-2013 "I think Wiki is better than a shared folder, where you have got to give someone access and the Wiki has more of an audit trail as well in terms of when you make some changes vs. someone send me file somewhere."

According to one of the client BAs both the team collaboration and issue tracking software were mainly used by client's local team members. The term wiki was used for team collaboration platform (both *JIRA* and *Confluence*). However, these technologies were also used by the vendor team (although not as much as the client team) for collaboration as seen in the next section "*JIRA* and *Confluence* as Knowledge Sharing Technologies"

FRNZ-ISE-06-2013 "I would say the collaboration tools have been more for our internal use…all the specs we do are on the wiki, minutes of meeting are up there, user stories, issues. Within the project team that's where we put stuff. The technical documentation is on there too. So I like it, it is a good resource, it's a good place to keep things, everybody knows it's the definitive place for stuff…We don't rely on just one thing but that has been something interesting and different thing for this project…it was pretty good".

4.8.1.1 JIRA and Confluence as Knowledge Sharing Technologies

Both *Confluence* and *JIRA* played a crucial role in team collaboration and knowledge sharing among team members from the client side. Project related information was stored in the project wiki *(Confluence)* in a directory-like format. Client team members (BAs, Project Manager, Tester) extensively used this platform to share, store, update and record information and discuss information for example bugs, e.g. IG-2091 17 listed in *JIRA* had around one hundred updates (mainly from the client team members such as BAs and PM).

[17] Project Wiki: Dashboard Issues IG2091 Date 31 Dec 2014

Members from the vendor team (BA and Developers) were also noted to be using *JIRA* but the frequency of usage was not high.

There were only two developers and one vendor BA listed in the project wiki. The usage of the project wiki (Confluence) of the developers is compared with two client side team members (a BA and the PM) shown in Table 4.7:

Role	Posts/Updates
Dev 1 (Vendor)	139 in 3.5 years (2011-2014)
Dev 2 (Vendor)	2 in 3.5 years (2011-2014)
BA (Vendor)	61 in 3.5 years (2011-2014)
BA 1 (Client)	120 in 1 year (2011)
BA 2 (Client)	240 in 17 days (2014)
PM (Client)	290 in 4 months approx.(2014)

Table 4. 7 Usage of JIRA by Different Roles across Client and Vendor Sites

Although the product owner and other steering committee members had access to the project wiki they did not use it as much as the other roles mentioned above (only one comment each found in the project Wiki).

The Vendor BA was an active participant in project wiki and used the platform to refine requirements, ask questions, make clarifications and discuss issues in the initial phases of the project start. Posts and updates on the project wiki from the vendor BA stopped after 2012 however the communication and collaboration continued outside the wiki through emails and teleconferences.

Project Wiki also offered a kind of support that the BAs could go back to see what the previous BAs have done. On the question regarding the most supportive and important technology, another BA answered:

FRNZ-CKE-05-2013 "Probably wiki because it has got the spec, in terms of knowing what previous BA have done I think now if you actually ask for what will be the most important , so I would say wiki."

Interestingly she highlights the fact that the practice of storage, use and grouping has improved over the time.

FRNZ-CKE-05-2013 "Our defects and stories, the specifications were a little bit out of post but I think they are being grouped so we have not got one page with links to all specs so I think that has been an improvement."

Furthermore, these collaboration technologies afforded some managerial outputs as well, in terms of status and progress reporting which was done through the available features

in the software. The project management team used the reporting features to update the steering committee regarding the status and progress of the project.

Answering a question regarding the need of additional collaboration tools one of the BAs said that there was no need for more and different tools. He suggested that the effectiveness of the existing tools could be made better if they were used by both sides. It is important to note that this BA was working with Dev 2 (Table 4.6) and made only two posts in the project wiki since the start of the project until the end of Dec 2014.

FRNZ-CKE-05-2013 "I don't think so. I would like that the developer actually looked at JIRA"

Although project wiki (*Confluence*) and bug tracking tool (*JIRA*) were considered helpful by the client project management team, however this view was not held by everyone in the team.

4.8.2 Hindering Role of JIRA and Confluence

Tool usage differed between the vendor and the client teams. Both tools (*Confluence* and *JIRA*) were mainly used among the client project management team for internal collaboration and communication purposes. The client specified the following challenges with the collaborative technology used in the project.

4.8.2.1 Complexity and Volume of Information in JIRA and Confluence

The client team members found the load of available information and how it was structured on both Confluence and JIRA as quite challenging

FRNZ-LEC-05-2013 "I think it is just [its] complexity, its [project wiki's] sheer volume of information…It is proving a little bit cumbersome and I think the BAs have reverted to use email with files directly"

The way these collaboration technologies were set up was not easy to figure out for certain team members especially for those who had limited knowledge about the tools. Even the researcher himself found the non-trivial structure of Confluence and how the information was stored overwhelming to deal with.

4.8.2.2 Challenging Interface of Collaboration Technologies (JIRA and Confluence)

On the client side some of the stakeholders found the interface of the project Wiki (Confluence) confusing and difficult to use. They considered, it had too much information to sift through. The information overload kept them from going back to the wiki to look

for the desired information regarding user stories, comments, and feedback on requirements.

SRNZ-MOL-12-2013 I am not an expert on JIRA so that is not JIRAs fault. I find the user interface very confusing but if I did more of it and used more of it I am sure it would be ok. When I do get in there [JIRA] it is kind of 'weary', I can put my issues in there, I can find them but I don't quite know how do they manage to design that".

In the review of sprint 1 the challenges with the interface, clarity of writing tasks and their dependencies was noted.

"•JIRA is challenging

•Need clarity in writing tasks

•Need to understand dependencies and this can be difficult to establish" [18]

4.8.3 Helpful Role of Spreadsheets as Collaborative Technologies

Spreadsheets were a preferred collaboration technology for the vendor team to specify and communicate and collaborate over requirements and changes. The vendor considered that the user stories were too high level to be translated into detailed specifications. Therefore, the vendor supplied the client team with the templates to use for specification The client BAs also found spreadsheets useful in sharing specifications once they started using them. The client realized that spreadsheets were not only preferred by the vendor but they also had some benefits hence they started using spreadsheets to specify requirements and changes instead of other type of artefacts for specification.

FRNZ-ISE-06-2013 "What we have found very effective is that there are supporting spreadsheet of access rights, screen mock-ups with annotations on them. They[developers] like that, because they know exactly is required it is not a separate bit of documentation, they look at that. It is very 'immediate' for them."

Similarly, for the vendor using spreadsheets was a logical way for requirements communication in a logical formal which was then easy for them to transfer directly into their system.

FRNZ-AVA-06-2013 "we ask the customer to document as much in a logical form and in a table-driven form using Excel, and we give them structures for that up to an extent that they can be logically communicated through excel, the tighter it follows our formats the more directly transformable it is directly to our system."

The vendor team confirmed that these Excel sheets were being used quite successfully to translate user requirements and changes into their system. According to them it was

[18] Project Wiki

possible to capture 80 to 90% of the business valued desired by the client but that depended on how good the client was in terms of representing their needs and specifying them clearly in those excel sheets.

FRNZ-AVA-06-2013 "we can transform out of those excel files [depending on] how good the customer is, 80 to 90% of the business value. And we do that with great success, and we only look at word documents with screen shots and mock ups for supplemental support of something that otherwise should clearly be documented in logic in an excel sheet."

During synchronous collaborations, spreadsheets provided the flexibility to the cross-site team members to modify the available content and even the spreadsheet structure to perform requirements engineering. Figure 4.19 captures the asynchronous use of spreadsheet by commenting and exchanging specification through emails as well as synchronous editing (simultaneously modification) done using GoToMeetings screen sharing feature.

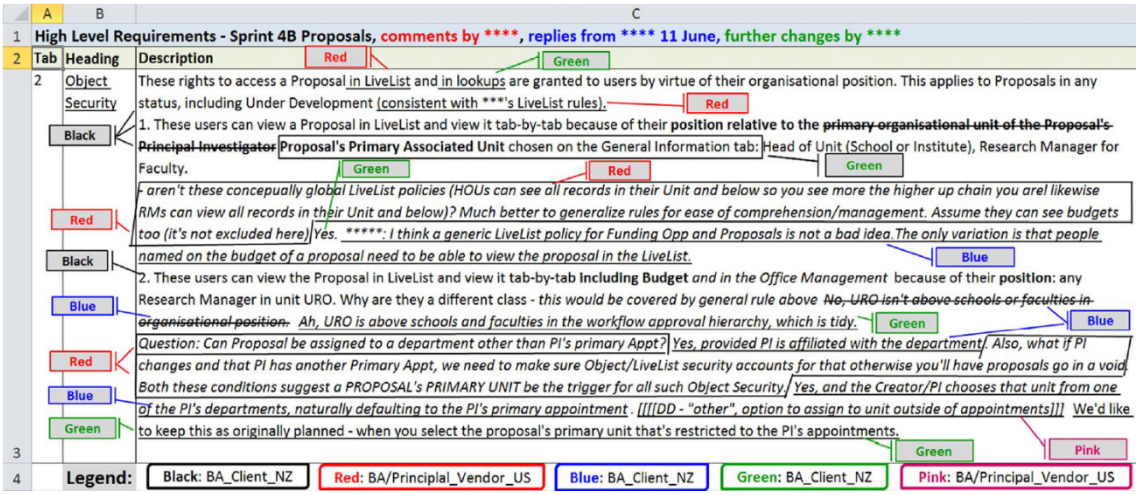

Figure 4. 19 Collaboration over Specifications in a Synchronous Web Conference Session

4.8.4 Hindering Role of Spreadsheets as Collaborative Technologies

Although the client agreed that spreadsheets were beneficial for collaboration however in some cases the use of spreadsheets was challenging for them as well. For the client team, it added an overhead of maintaining two mechanisms to perform requirements related tasks which often resulted in duplication of effort. For example, it was not a preferred tool for communicating requirements and bugs since the client team was already using another collaboration technology (JIRA) for issue or change related communication. Although the vendor did not have any tracking system of their own but according to the client they could not be persuaded to use *JIRA*. This meant that the collaboration technologies such as *JIRA* and *Confluence* were mainly for internal collaboration and defect reporting as reported by client team members.

The client had to use spreadsheets for reporting issues and bugs to the vendor, which had links back to JIRA incase the developers needed more information regarding those issues. As reported by the client, the developers use *JIRA* only if they wanted more details about a particular issue that was stored in *JIRA*. However, these collaborative technologies were not often utilized by the development team as seen below:

FRNZ-CKE-05-2013 "I have to send them[issues] on a spreadsheet which they can then click on and go into the defects but they just don't like looking at it [JIRA]."

The vendor however, required the client to use spreadsheets mechanism to capture requirements, changes and issues. Since spreadsheets were their preferred technology for collaboration, they only looked at other places for information only if such information was not made available by the client in spreadsheets.

FRNZ-AVA-06-2013 "Because we don't want anything in a word document in narrative that adds requirement that's not already specified. The only thing that a word document or PowerPoint or screen shot or mock up should ever do is complement what we are getting in some relative form of logic , where we are very explicit with [the client] saying do not burry requirements in documents to make us try and find them. That won't be a very successful game for either party. We will look to the spreadsheets and a logically tabbed structure a self-referential structure to give us their requirements, then and only then can we get the business to be on a co-owner of those requirements."

However, for some client team members having to deal with two mechanisms (Project Wiki as well as Excel spreadsheets) to manage requirements as well as issues/bugs was too challenging. One of the client team members stopped putting work on Project Wiki to avoid duplication of effort in later stages of the project.

SRNZ-MOL-12-2013 "we are saying we want this thing fixed, here are spreadsheets. If there is an issue, we may have logged it in JIRA but we have given up on saying please look in JIRA."

The statements above also suggest that the collaborative tools (*JIRA* and *Confluence*) many not have been adequately configured for sharing knowledge, related to requirements and team collaboration purposes and therefore were not being properly utilized to their potential by the client and vendor teams.

4.8.5 Helpful Role of Email as a Collaborative Technology

Email played a vital role in collaboration and communication not only among local team members on the client side but also with the remote team members for sharing requirements and changes to obtain their feedback. They were used as a mechanism to communicate and collaborate 'quickly' within the team as well as with the vendor:

FRNZ-ISE-06-2013 "Quick conversation you go on skype, you send an email".

Messages and excerpts and attachments from emails were consistently cited on project wiki for discussions regarding requirements and software bug related discussions. For example, one of the client team members noted

FRNZ-ISE-06-2013 "[The vendor] introduced a Workflow spreadsheet in an online meeting on 5 Feb 2013 and sent through [email] the attached sample"

It provided a record of communications and the possibility to search and refer back to the old versions of documents shared between the team members. On the question of regarding the most useful technology one of the respondents replied:

FRNZ-CKE-05-2013 "I think if you took wiki out you would replace it with folder structure…defects you can manage them in Excel, so you could do without….email would be tricky, because you are sending screenshots so email would certainly be tricky."

Similarly, one of the interviewees said that the BAs are reverting back to emails for communication as they feel information overload in the project wiki.

FRNZ-LEC-05-2013 "I think the BAs have reverted to use email with files directly"

Emails were used for requirements and change management related activities for example during *scope signaling;* where initial drafts of candidate requirements were shared with the vendor for his review and viability in the coming sprint,

Emails with spreadsheets were also used to share the initial sprint requirements for further elicitation analysis, scope negotiations. Emails with modified and revised and updated specifications contained both in text as well as in attached spreadsheets were continuously utilized until specifications were (see excerpt of email exchange with colored text from cross site BAs. Image size reduced for information privacy). [16]

The client frequently used emails for exchanging information, requirements related activities and as collaborative technology as can be seen in the following statements

FRNZ-AVA-06-2013 "[changes to baseline] was called out in many many many email exchanges".

FRNZ-AVA-06-2013 "because our software is very easy to configure and customize there were many many many many discussions which again commemorated in many many email exchanges."

The vendor further suggested that using email as a collaborative technology in combination with other technologies for synchronous communication (*Skype* or *GoToMeeting*) was seen as a viable and adequate mechanism to carry out collaboration activities both for collocated as well as global software development projects.

FRNZ-AVA-06-2013 "I would also argue … that 90% of the communication even on the same floor happen by Skype or email or other means. So I think that the world we live in that the whole notion that, if people are doing that between those cubicles they are actually and finding it effective mechanism it that way. And that is exactly just as viable globally".

4.8.6 Hindering role of Emails as a collaborative technology

One of the client team members suggested that not only emails could be interpreted wrongly but in a GSD project fixing the issues created by miscommunication through email was very hard.

SRNZ-MOL-12-2013 "you send someone an email, you are thinking that you have written something 'here' and they may read the email and end up 'there'. It is not what you meant but you have to be so careful wording things. When there is distance and time zone difference and you can't just pick up the phone and say 'what did you mean here?', it is very hard, very hard.

One of the BAs highlighted the fact that email communication could be misinterpreted

SRNZ-MOL-12-2013 "Because trying to do that over the email, again the potential for misunderstanding when you have email is greater when you have a face to face conversation."

Another problem suggested by the client team members was that they did not think emails were good for carrying out negotiations:

FRNZ-ISE-06-2013 "But [email] is not a good place for negotiation. If you are negotiating, you need to be much more face to face [conversation]".

However, during the course of the project email as a collaborative technology was extensively used and considered vital as previously reported and also observed in the project wiki where emails are referred to on many occasions.

4.8.7 Helpful Role of Web Conference as Collaborative Technology

Both client and the vendor team members agreed on Skype and GoToMeetings being helpful collaborative technologies and among the main tools used for collaboration. The ability of these synchronous tools to share screen was considered most helpful by the geographically distributed team members across sites. Consider the following statements from the client team members:

FRNZ-ICK-05-2013 "I always like Skype and GoToMeeting where you can actually work collaboratively with someone who is remote is extremely valuable"

Screen sharing feature in conferencing tools such as (Skype and GoToMeetings) was found to be of great value by the team members. Screen sharing was one of the most commonly used mechanisms when collaborating over requirements changes.

FRNZ-ICK-05-2013 … and I think [both] are quite good tools especially because you can put some screen up and they can show you something or you can show them something…. rather than trying to describe it to them in words, for them to be able to actually see it and work through some defect that you have found, show them, and they can actually see the problem"

Finalization of the specification handover process was done through synchronous meetings through audio conference calls (Skype and GoToMeeting) and utilizing screen sharing facility of the conferencing tools. The specification handover (for requirements and changes) included almost all of the RE activities such as scope negotiations, clarification, confirmation, elicitation, analysis and finalization of the specifications.

FRNZ-MOL-05-2013 "With GoToMeeting one can be looking at a screen together and interacting with it together or looking at a spreadsheet together, obviously the Google docs type of thing one can be working simultaneously on a document and typing in what comes in so the whole thing where you are recording minutes of meeting, so much easier in those tools. So they make a huge difference"

The vendor talked about the importance of sharing screens and the fact that this technology was generally very effective and helpful apart from the rare technical issues. The following quotations discuss the nature of the collaborative tools and their support in remote collaboration

FRNZ-AVA-06-2013 "[Skype and GoToMeeting] have a significant overlap, Skype is a more informal mechanism and GoToMeeting is a more formal mechanism. GoToMeeting lends itself better towards multi-party screen share. … those vehicles are absolutely 100 percent fine in fact they are fantastic."

According to the vendor the reason to select Skype and GoToMeetings as collaborative technologies was their 'immediacy' and learnability.

FRNZ-AVA-06-2013. We choose to have a minimum overhead, nobody needs to be taught how to use linked in, nobody has to be taught how to use Skype or GoToMeeting they are immediate tools, and make us have an immediate connection to other parties and a direct one.

These technologies were considered so effective by the same respondent that the distance impact was considered negligible.

FRNZ-AVA-06-2013 "We are talking through text, verbal [voice] and shared screens; we are collaborating as if we were adjacent to each other".

In comparison with the video conference the client team members preferred audio conference. This is quite surprising because most of the literature talks about the desire of having video conferencing as the richest media second only to a face to face meeting (Klitmøller & Lauring, 2013).

FRNZ-LEC-05-2013 "I think the audio conferencing is probably the most effective, so long as the quality of sound is good. Mainly because we can communicate amongst each other by signals, by pointing the

things on bits of paper, and by passing messages, so that we say that we don't agree on the approach or response while the teleconference is on."

The client team members suggested that for carrying the requirements related discussion forward and not getting stuck, the audio conference was considered more effective and less rude. It allowed the messages to be shared with the local team without letting the remote party know about it.

*FRNZ-LEC-05-2013 "So I can look round and I can (*does a 'cut-throat expression) tell the people [that] I am going to 'cut him off' [*Laughs] … thanks [remote team member], we've done that, can we just move on to the next point".*

4.8.8 Hindering Role of Skype or GoToMeetings

There were two main problems with the teleconferences using Skype or GoToMeeting which were poor audio quality and voice latency as discussed below:

4.8.8.1 Poor Audio Quality

Poor audio quality during the long synchronous meetings made it difficult for some of the participants to listen to and understand the discussion which was considered very stressful by some of the participants.

FRNZ-LEC-05-2013 the technology is not ideal and often it is very difficult to understand, the audio quality is poor which makes it very very stressful and tiring to listen.

FRNZ-ICK-05-2013 "My impression is that GoToMeeting isn't quite as bad as Skype so and also with Skype the voice quality seems to suffer when lots of parties on a call"

The long duration of the teleconference calls coupled with poor audio quality was difficult to manage especially at crucial time of setting contract conditions as expressed below

FRNZ-LEC-05-2013 "probably the most difficult thing I faced, was the initial setting out of the contract conditions…I had to join [the PM]in teleconference with the vendors that were up to three hours, they were absolutely exhausting physically… I felt that I was signing the [client] up to something which is worth several million dollars, setting the future for my office for research, and I was struggling to hear and understand what was being said…just hearing what he was saying, and understanding what he was trying to tell us was very difficult".

In one instance poor voice quality and latency caused so much misunderstanding and frustration to build that one of the client and vendor teleconference had to be abandoned.

4.8.8.2 Voice Latency and Echo

The latency in voice was considered a very big challenge by the client BAs during requirements related conversations with the vendor that caused misunderstanding, miscommunication and frustration.

FRNZ-MOL-05-2013 Latency on the voice side of GoTo meeting, absolute killer, when you are trying to discuss concepts you are trying to understand each other, you have only got a voice…. and you are talking on top of each other, because there is this 500 mille second latency in the voice call, that's a real killer"

Even with the screen sharing feature which was considered most useful by almost all of the participants the delay and latency was one of the reasons for participants of a conference call.

FRNZ-ICK-05-2013 "with Skype often there is quite a lag between the time you move the mouse on screen locally and they see it move, it can be quite 'marked', one or two seconds so that can be quite annoying.

Similarly, another participant on the client site considered the use of screen sharing and synchronous meeting useful but found the echo over voice quite awkward and complained about the overall quality of the technology as not being good.

FRNZ-ISE-06-2013 "a little bit of echo over voice is a bit awkward during conversations while using Skype and GoToMeeting. The delay is a little bit awkward…the technology is not up to 'really good quality'"

In one instance the client BA was so frustrated that he decided to make a direct phone call and discuss the issue rather than going through the GoToMeeting session.

FRNZ-MOL-05-2013 "…so in the last couple of weeks I have taken, just to ring in the guy direct, even that's not a land line at his end, so he's got half the latency at his end. I haven't got any on my end but you know that's very hard."

One of the client BAs suggested if the vendor used a headset the audio quality could have improved which suggests that the technology was not optimally utilized in some cases by team members in the CICP project.

FRNZ-ISE-06-2013 "I think if [vendor] had a 'headset' that might have actually helped. We might hear [the vendor] a lot better. I don't know what he was using earlier, but was hearing so badly. Everybody in the room was getting frustrated.

4.8.8.3 Preference of Using Non Rich Media for Communication

At the start of the project, client and vendor team members were using video conferencing for requirements clarification but they stopped using video conferencing. Respondents expressed disliking of video conferencing for cross site requirements and project related team meeting and preferred audio conference. They found it difficult to 'pose' well in front of the camera for the whole duration of the conference.

FRNZ-LEC-05-2013 "Keeping your expression looking good for a two-hour meeting where half of it is irrelevant is really difficult."

Some roles, such as business analysts preferred the use of audio conference as well as they were not comfortable sitting in front of the camera for long durations with people having a difficult personality to deal with.

FRNZ-CKE-05-2013 "In video conferencing I mean we don't really need to look at [remote project member] video conferencing so [laughs], that's ok [we don't need that].

Similarly some other participants from the client team did not like using teleconferences because of the possibility of prolonged conversation by particular individuals on the vendor side.

FRNZ-LEC-05-2013 "[the BAs] do Skype from time to time but when the [Vendor] goes off on a tangent we want to role our eyes. It is best not to be using Skype to be quite honest."

Table 4.8 summarizes the (hindering and helpful) role of collaborative technologies.

Collaborative Technology	Helpful Role	Hindering Role
JIRA (Issue Tracking Software)	• Provided issue/ bug tracking facility. • Helped in testing and issue status reporting. • Provided management support for project monitoring. • Helped in project status reporting. • Facilitated issue related collaboration. • Facilitated information retrieval across sites. • Provided issues related collaboration information for knowledge workers. • Helped in training new team members. • Helped in reflection on the previously logged issues and for future planning.	Lack of interest of using the tool from the vendor team. Challenging interface to discourage effective usage. Huge volume of information and complexity to effectively utilize the information. Variable usage (from high to low or inactive) both from the client and vendor team members which discouraged effective collaboration.
Confluence (Project Wiki/ Team	• Helped in organizing, storing and sharing project related artefact and knowledge in an organized way.	Lack of interest in using the tool from the vendor team.

Collaboration Software)	• Helped as an information retrieval facility available for cross site communication and usage. • Facilitated sharing and storing product documentation and vendor information. • Helped in storing and sharing end user documentation. • Provided consistent documentation of the multiple versions of sprint requirements specification.	Challenging interface to discourage effective usage. Huge volume of information and complexity to effectively utilize the information. Variable usage (from high to low or inactive) both from the client and vendor team members which discouraged effective collaboration.
Email	• One of the most relied and heavily used asynchronous collaborative technologies. • Extensively used as a reliable tool to share requirements and change related artefacts (specifications and design) and feedback on artefacts. • Containers and carriers (as attachments) of requirements and change related documentation and feedback/clarification).	Had the downside of being misinterpreted. Not a good technology to carry out negotiation. Difficult to keep track of various threads of communication.
Web Conference	• Facilitated distributed collaboration in an affordable manner. • Favoured due to their immediacy and learnability. • Facilitated various requirements related synchronous meetings for review, analysis, design, specification and validation activities. • Found most effective when used with the screen sharing feature that allowed. sharing of requirements artefacts (e.g. spreadsheets) and their synchronous editing. • Acted as a preferred media for communication for negotiation and resolving conflicts • Considered an attractive.	Poor audio quality affecting the usefulness of communication. Voice echo hindering the quality of communication. Voice latency causing people to talk over each other and missing important conversation. Preference of using less rich media for communication due to lack of bandwidth and poor cross site relationship.
Spreadsheets	• Acted as a bridging function between the core development team and other distributed stakeholders (business analysts, subject matter experts and project managers). • Supported all requirements related activities while serving a summarizing and communicative across distributed stakeholder groups. • Complemented special purpose collaboration technologies (e.g. ticket tracking systems, IDEs etc.) to cater for the distributed collaboration needs. • Enabled discussion and supported synchronous editing when used in	Having to use template from the vendor. Duplication of effort. Proliferation of isolated and duplicated data pools (bug report, design docs, specifications, workflows). Adding the overhead of maintaining two systems or managing requirements and changes in parallel.

	combination with as Lync and GoToMeeting with active screen sharing feature. • Allowed asynchronous use of spreadsheet use was supported through emailing files between sites. • Used as a web based artefact repository sharable as attached files with embedded links to the bug-tracking tool (JIRA).	

Table 4. 8 Summary of the Hindering and Helpful Role of Collaborative Technologies Used in CICP

4.9 Chapter Summary

In summary, this chapter provided a background and presented a contextual analysis of the case studied. The prescribed and practiced models for requirements and change management in the selected project were analyzed and the associated challenges were discussed in detail. The model in practice for the studied case was analyzed based on the lifecycle activities of a requirements change and the challenges related to each of those activities are described. There were added challenges due to the globally distributed nature of product development in this case that were also reported and discussed. Global distance with its three dimensions (geographical, temporal and socio-cultural) was found to be the cause of these additional problems. The challenges due to distance aggravated the existing challenges faced by the practitioners in managing requirements. Technology was found to have played a major role in making the distributed development possible. During the course of the studied project collaborative technologies used by the practitioners were generally found to be helpful and adequate however, they did throw some challenges at the stakeholders who were using them. These challenges were also discussed.

The next chapter will analyze the second case in a similar fashion as done here and both this chapter and the next will be used to do a cross case comparison in chapter 6.

Chapter 5. Analysis and Findings- Case 2

This chapter reports the analysis and findings from Case 2 of the two selected cases in this research. Sections 5.1 and 5.2 provide the overview and context of the Case 2 project and Section 5.3 presents the project studied through an activity theory based model which was considered useful in understanding and analysing collaborative work for Information System (IS) development. Section 5.4 and 5.5 provide a high level overview and analysis of the prescribed requirements change management model respectively. Section 5.6 and Section 5.7 then present the analysis and findings of the requirements management model in practice based on the lifecycle activities of requirements change and discusses the challenges faced while performing these activities. Then section 5.8 presents the challenges faced by the practitioners in managing requirements change in a globally distributed environment. Finally, in Section 5.9 the helpful and hindering role of collaborative technologies is presented. Finally, Section 5.10 summarizes and concludes this chapter. The findings presented in this chapter complement the findings of the previous chapter and provide the foundation for the cross case analysis and discussion reported in chapter 6.

5.1 Overview of the WIPS Project - Case 2

The primary objective of this project was to develop a *web interface program* for an information management system (IMS) and to integrate the WIPS interface with the existing online tools linked with the current website. The new interface was to provide better data manipulation and visualization and to maximize user understanding of the available information (FBO.Gov, 2011).

The selected project at GSD Inc. was being carried out between a US based governmental client and a US based vendor who had an offshore development site in Pakistan. The *client* was an agency within the U.S. Department of Transportation that supports state and local governments in the design, manufacturing and maintenance of local infrastructure and construction. The *vendor* of this project was a US based contractor for software development (fictitiously called) GSD Inc. Founded in 1999, GSD Inc. is a privately owned, ISO and CMMI Level–II certified global information-engineering firm. This small to medium sized organization has almost 50 employees, out of which 45 employees worked at the offshore development site in Islamabad Pakistan while the remaining staff operated from the head office in Virginia, USA. GSD Inc. mainly provides offshore software development services to four clients in the USA.

There were several reasons why GSD Inc. provided an appropriate representative site for this study. Firstly, there was a trusted relationship between the researcher and the company due to a previous successful research collaboration that was carried out for the researcher's master thesis in 2010. Secondly, the organization has been involved in outsourced software development work for almost fifteen years. Also the nature of the project under investigation was deemed fit for the GSD criteria laid out for this study. Thirdly, GSD Inc.'s offshore development model was a typical reflection of many regional small to medium sized software development organizations in Pakistan.

Most of these regional software development organizations had at least one main offshore development site in Pakistan with an (onsite) client coordinator and limited staff (in the USA or other countries in the West). The researcher himself has been associated with two IT outsourcing companies for over 5 years located at one of the technology parks established by Pakistan Software Export Board (PSEB, 2015) in Islamabad and is personally aware of at least ten companies with similar set ups working in the same location and around the city.

A slight variation in the working style of GSD Inc. was that the business analyst from the vendor played a bridging role and liaised with the actual client but at the same time acted as a client proxy for the offshore development site in Pakistan. Such roles are known as information brokers, liaisons, gatekeepers who act as boundary spanners that fill the structural holes of social networks in distributed organizations (Boden & Avram 2009). Often these boundary spanning activities are carried out by BAs who can understand and communicate in the business domain but are also well versed in technology (Nasir & Hameed 2004).

The context of the WIPS project is explored and presented here to provide a broad background for comparative understanding and analysis across the two studied cases as discussed previously in Chapter 4 Section 4.1.

5.1.1 Need for the Project

This project is part of a long term strategic research program carried out by one of the US federal government's state departments for understanding, managing and forecasting the performance of construction related raw material. Over the past 20 years the research program accumulated a huge amount of data, a mass of documentation as well as related tools which comprise the existing IMS (FBO.Gov, 2011). The available data was seen as potential information waiting to be explored, analysed and converted into useful knowledge by means of the WIPS system to be developed. There was a stated need to

"improve public access and understanding of [the existing system's] data" (FBO.Gov, 2011) and to keep up with the current information technology.

The newly developed web interface would replace the functionalities provided by an existing website.

5.1.2 Contextual Factors and Dimensions

A summary of the contextual factors of WIPS as a globally distributed development project is provided in Table 5. 1 as done previously in Chapter 4 Section 4.1.2 for the CICP project.

Factors and Dimensions	**Detail**
Market	
Industry	**Road Construction**
Sector	**Government**
Product	
Maturity	Custom Development/COTS Extension
Software	Information Management System
Application Type	Web Application
Size	Medium
Cost	USD $3.5 Million
Complexity	Medium
Requirements Change Rate (Volatility)	Low to Medium
Development Organization / Vendor	
Maturity/Certification	Mature-CMMI Level 2 Certified
Team Size	Ten
Development Process	
Outsourcing	Far-shore Outsourcing
Methodology	Waterfall by Feature
Workflow	Sequential by Feature
Practices	Prototyping, Incremental, User Stories
Global Collaboration	
Collaborating Units	Pakistan, USA
Collaboration Technologies and Tools	Email, Microsoft Lync, Bug Tracker, Design Specification, Spreadsheets
Collaboration Artefacts	Contract, Web UI Prototypes, Enhancement on Design Specifications, Software Releases, Requirements Change Log
People	
Client Collaboration Roles	Manager, Product Owner, Business Analysts, Tester

Client Governance Roles	Project Manager, Domain Experts, IT specialists, End User Group Members
Vendor Collaboration Roles	BA/ Architect/Principal
Vendor Governance Roles	Development Manager, Project Manager, Team Leads
Project	
Status[19]	Finished: Product delivered to the client
Outcome	Success- Satisfied client and end users, considered a success by the client

Table 5. 1 Project Context- Software Engineering Adapted From Hussain & Clear (2014)

5.2 Contextual Analysis the WIPS Project Using ABC Framework

The contextual depiction and analysis of the WIPS project is based on the Activity Theory Based Computing (ABC) Framework provided by (Korpela et al., 2002) as discussed in Chapter 4 Section 4.2.1 and 4.2.2. The terminologies and concepts of Activity Theory analysis are again used here again to provide the reader with a mechanism to aid the understanding of the elements of a work activity in a single snapshot. The following sections discuss various work process elements and activities such as actors, instruments and means of communication in a geographically distributed collaboration. As discussed in Chapter 4 Section 4.2.2, mapping of work activity elements is done on the basis of information collected and analysed from interview data and the available project artefacts for the WIPS project. The interview questions that allowed this mapping related to: sites and roles of participants in the project, communication and collaboration technology used, development processes.

5.3 Mapping CICP Work Activity Elements Using ABC Framework

An example activity from the WIPS project i.e., converting design specifications into a web application is mapped on to ABC framework using its various concepts and work elements. The mapped work activity elements of Activity Theory include: *Object, Outcome, Actors (and Sites), Means of Communication and Coordination, Means of Work/Mediating Instruments, Mode of Operation* (Korpela et al., 2002). The mapping was carried out the basis of information collected from the interview data and the available project artefacts.

As done for Case 1, in Chapter 4, the work activity and its elements are presented here using an abstract visualization for a broader understanding and analysis of the project

[19]Status reported here is subsequent to the study i.e., at the time of finalizing the thesis. Source of information: Several newsletters published online by the USA Govt. agency involved as the client which report on the successful completion of the project.

work. This project level view complements a detailed analysis of requirements and change lifecycle activities is carried out later in Section later 5.6.

5.3.1 Object and Outcome

For WIPS project an example work activity is mapped using ABC framework which takes design specifications as a shared object for this activity. Through collaboration of people (a collective activity) following a predefined process this shared object is converted into an outcome i.e. a web application in this case.

5.3.2 Mode of Operation

The mode of operation for carrying out this shared activity is both Collaboration Driven as well as Technology Driven (Korpela et al., 2002). It means that the shared object is converted into an outcome through collaboration carried out using various tools and technologies.

5.3.3 Actors and Sites

The project was carried out in an environment where the client and the vendor's development team were globally distributed. Requirements development and implementation activities for this project were coordinated over geographic, temporal and cultural distance. There was a ten-hour time difference between the client and vendor's development site.

The vendor's offshore development team is shown enclosed in the large box on the left hand side (Figure 5. 1). It consists of a Development Manager (DM) who manages the project and development from the Pakistan site, a Development team lead (TL) involved in the architecture and design of the product and also in the supervision of further downstream development tasks, two Developers (Dev) involved in various software development activities related to the project as assigned, a Document Engineer (DE) responsible for the design documents, a Graphic Artist (GA) involved in GUI and prototype development of the web interface, a QA Manager (QAM) and Testers responsible for internal quality assurance and testing of the software.

Additionally, the main actor at the onshore client coordinator site (shown in red box) include the Proxy client/ BA. The lines and arrows depict communication flow and the degree of communication; solid lines show more frequent (or intensive) communication relationships and dotted lines show weaker (less frequent) communication.

The key stakeholders from the client side were the Project Manager (in Purple), Domain Experts (in Blue), Contractors / Partners (in Grey) and selected existing users (in Yellow).

The colours represent different remote locations (within the USA) from which these actors participated in the project.

The vendor's BA/proxy client collaborated with stakeholders from the client agencies including its ICT staff, the Contracting Agency's Technical Representative (COTR), existing product Technical support service vendor team, development team, technical review board committee members and existing users of the product. The roles were involved in collaboration for both requirements and change development and user acceptance testing. These actors perform activities based on their assigned roles and the division of labour.

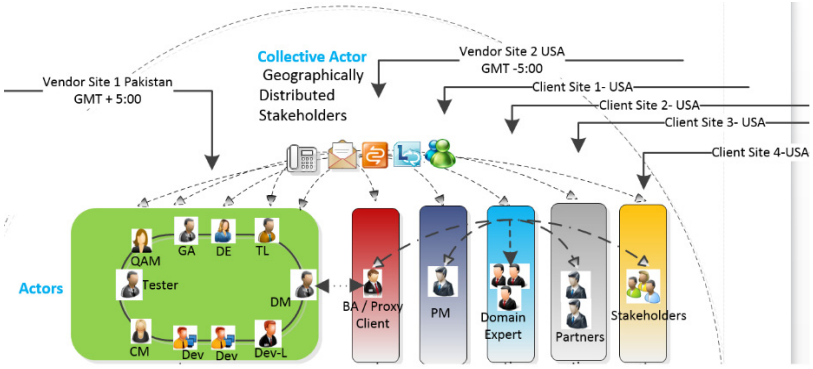

Figure 5. 1 Geographically Distributed Sites and Actors in the WIPS Project

5.3.4 Means of Communication and Coordination

Actors utilized several means to communicate and collaborate on their work activities. For this project the tools used for daily *synchronous* communication across sites were Phone calls, MS Live Meeting (conference calls), MS Lync (Text / chatting software). These tools were supplemented by some application/ screen sharing software for work collaboration as required. Cross site *Asynchronous communications* took place through emails, an online ticketing tool, and a bug tracking system (developed in-house) called QAQC Centre. The actors used these tools to exchange requirements and change-related information, share resources, report bugs and verify implementation Figure 5. 2.

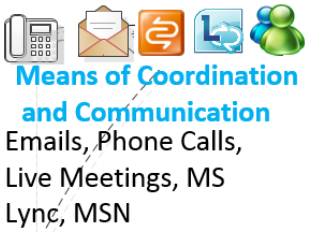

Figure 5. 2 Means of Coordination and Communication

According to the WIPS project contract at least two face to face meetings had to be conducted with the client stakeholders by the onsite vendor BA in the USA; a) the initial

project kick off meeting and b) a reporting meeting after the framework for the WIPS program was developed by the vendor (FBO.Gov, 2011). Face to face communication with the offshore development site was also possible through the proxy client's periodic visits (around every three months) to Pakistan. Remote interaction with the development site relied on the same communication channels and tools as utilized by other project stakeholders. No onsite visits were made by any development team members from Pakistan, either to the client organization or to the onshore vendor site.

Non-code artefacts e.g. requirements specification spreadsheets, design documents and the mock GUI were also used as a means of communication and collaboration. During the course of the project these mechanisms were employed by the actors for stakeholder interaction, requirements and change collaboration, release testing and verification, as well as for the release deployment activities.

5.3.5 Means of Work-Mediating Instruments

Actors applied various mediating instruments including their domain and contractual knowledge to convert design specifications into a release. The vendor and client team members implemented design specifications utilizing various instruments and means of work. In this case the mediating instruments included: modelling tools (MS Visio), graphic editing (MS Paint), project / code repositories (Visual Source Safe) and team collaboration software (MS Team Foundation Server). Other means of work included Bug Tracking and Ticketing software (QAQC), MS Excel spreadsheets, GUI mock ups. The work activities shown in Figure 5. 3 were carried out in a collaborative and technology driven mode. The overlap between some of the means of communication and the mediating instruments (such as requirements specification spreadsheets, GUI mock ups) indicates that these communication and collaboration artefacts were also used as the mediating instruments.

Figure 5. 3 Means of Work-Mediating Instruments

The tools used for carrying out requirements and change related collaborative work as identified through interview data and artefacts was shared with the offshore development organization for their verification and feedback.

5.3.6 Conversion of Design Specification into Web Application

The vendor development team implemented mutually agreed design specifications contained in a formal [*WIPS] Version 1.0 Design* document finalized after multiple iterations and signed off by the client. These documents included design specifications, requirements statements, workflow diagrams, annotated screenshots of the changes specified and screen mock ups developed to capture changes in requirements.

The requirements were implemented in an iterative fashion using the communication and collaboration means through the mediating tools in place. Once implemented, the software release was tested according to the design specifications, first by the internal testing team and then by the client through alpha and beta releases. Test and fix cycles continued until a particular release was stable and acceptable to the client (Figure 5. 4).

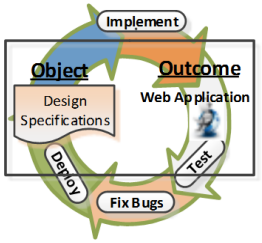

Figure 5. 4 Conversion of Design Specification into Web Application

Identified issues were communicated through the same channels as previously used for communicating and collaborating requirements activities however these issues were reported and tracked through ticketing software provided by the vendor.

Figure 5. 5 captures the complete activity in one snapshot for a quick visualization and understanding of the elements of work activities involved in the WIPS project.

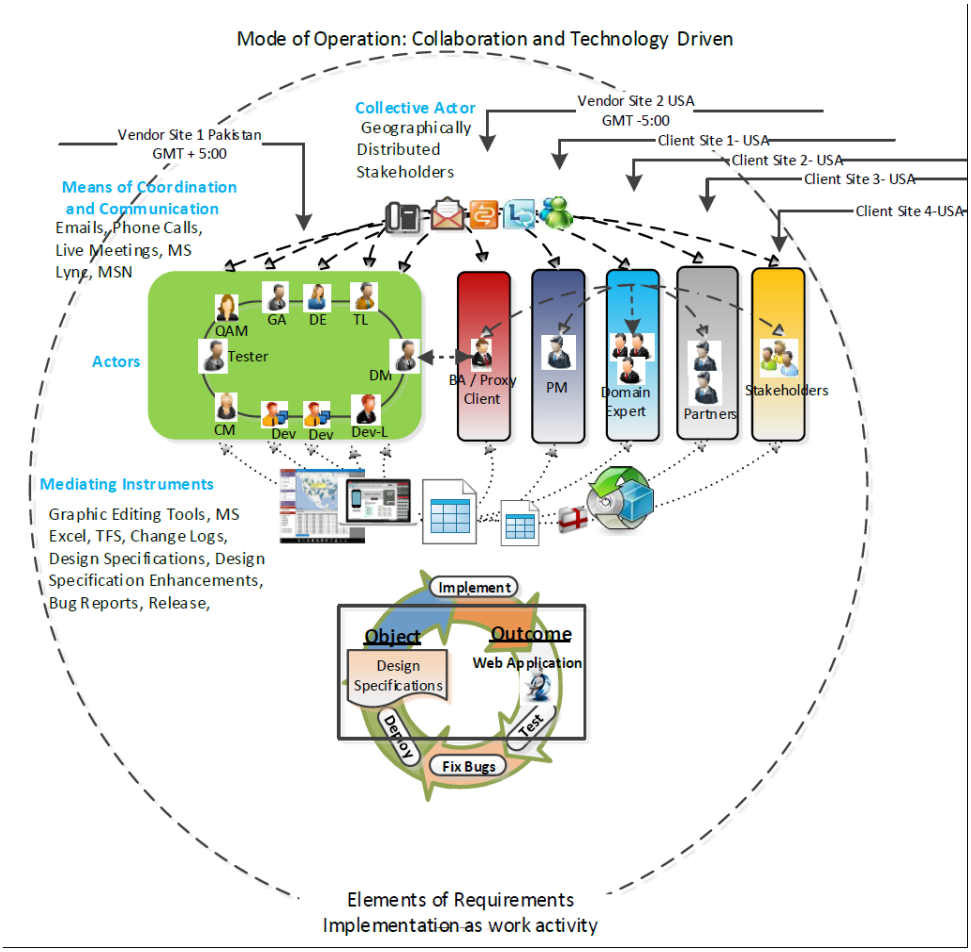

Figure 5. 5 ABC Based Model of the WIPS Project Showing Elements of Work Activity

5.4 Overview of the Prescribed RM Model in the WIPS Project

The studied organization followed practices based on CMMI Level II guidelines for their requirements management process (CMMI Product Team, 2006; Software-Quality-Assurance.org, 2015). The prescribed RM process model (Figure 5. 6) and the related information in this section is taken from the company's RM process documentation(GSD Inc., 2009). The prescribed RM process has three main elements that are *Roles, Activities* and *Artefacts*. The Software Manager (SM), Team Lead (TL) and Quality Assurance Manager (QAM) are the three roles involved in the process who perform the activities and produce related artefacts (Figure 5. 6). The main process activities shown in the model are 1) Obtain requirements, 2) Analyse requirements, 3) Obtain Commitments on Requirements, 4) manage Requirements Change, 5) Manage Bidirectional Traceability and 6) Identification of inconsistencies. In the studied company the degree to which the CMMI prescribed practices for managing requirements were followed differed from project to project.

The RM process was invoked when some requirements related reference material is received by the organization (RFP document in this case). The process followed a sequence of activities and the people involved in these RM activities participated based on their prescribed roles. Each of the six activities mentioned above resulted in at least one requirements related artefact. The software manager (SM) was involved in carrying out all six process activities described above and was the only contributor in producing the reference materials, documents and RM workbook. The Team Lead (TL) was involved in all activities except in 'Obtain Requirements' activity. The TL and SM work on analysing requirements and producing the results in requirements management workbook.

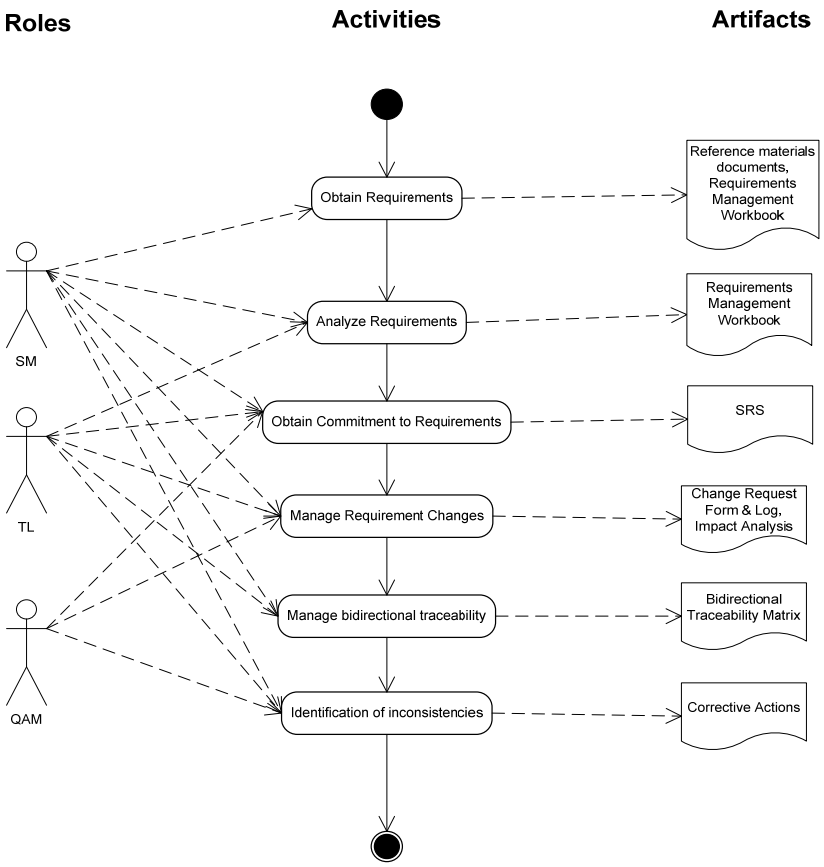

Figure 5. 6 Requirements Management Process Model (Activity Diagram)

5.5 Analysis of the Prescribed RM Model in the WIPS Project

Upon analysis several inconsistencies were identified and observations were made regarding the prescribed RM process model, contribution of roles activities and requirement related artefacts. These observations are discussed in the coming sub sections.

5.5.1 Prescribed Process Model Related Observations

The model was not developed for global software development projects and therefore does not show distributed sites and roles. The model is linear in nature with only forward sequential activities (Figure 5.4) which does not reflect the cyclic, iterative activities of managing requirements in the studied (globally distributed) project. Based on the participant interviews it is realised that the model does not show some important RM practices such as change validation, negotiation and prioritization. Furthermore, this sequence of RM and traceability activities provided in the prescribed model differs from the sequence given in the RM process documentation. According to the RM process documentation traceability is to be managed once the commitment to the requirements is received from the client. This is in contrast to the sequence shown in the model where traceability starts after requirements change management activities are completed.

5.5.2 Roles and Contribution Inconsistencies Identified

Analysis of the collected data revealed inconsistencies between the prescribed model and the information collected through interviews and process documentation about the roles involved and their participation in RM activities. For example, requirements elicitation and analysis are shown to be performed by only one person (Figure 5. 6), however according to the process documentation at least five individuals (including the Project Manager) contributed in requirements elicitation and analysis activities (GSD Inc., 2012). For analysis and design phase, similarly, three more individuals were involved in addition to the two (TL and SM) shown in the model (GSD Inc., 2012).

Furthermore, the RM artefact analysis revealed that none of the roles (SM, TL and PM) specified in the prescribed model directly contributed to the RM workbook as shown in (Figure 5. 6) (GSD Inc., 2012a). Instead, the RM workbook was drafted by the QA department under the supervision of one of the QA managers (GSD Inc., 2012a) utilizing information from the design documents.

Similarly, in the prescribed model (Figure 5. 6) the SM, TL and QAM appear to have contributed in managing requirements change, however, the interview data revealed that the configuration manager, proxy client and three other team members were also involved.

5.5.3 Requirements Artefacts Produced

The artefacts produced by RM activities in the prescribed model (Figure 5. 6) and their nature also differed from those described in the process documentation. According to the RM process documentation (GSD Inc., 2009) a list of all requirements has to be produced

once all the reference material related to requirements was to be obtained. This list would eventually transform into one or more of the requirements related artefacts such as: System Requirements Specification, System Design Specification, Requirements Management Workbook, Design Documents, Prototype and Use Case Document. However, no such list of requirements was found during the artefact analysis.

The prescribed RM policy required all functional and non-functional requirements to be recorded in RM workbook (GSD Inc., 2012a) however only ninety-five requirements (estimated to be 20% of the total) were noted in the RM workbook. It resulted in a disconnect between the RM workbook, the initial high level requirements and the detailed design specifications. Furthermore, according to the prescribed practice changes in the design specification documents were to be formally managed through a Tailoring Request Template (TRT). However most of the changes made to the design documents were not formally approved, recorded or managed through the TRTs. Furthermore, according to the prescribed model, members of both the development and management team were responsible for creating change related documentation. However, the analysis of the RM documentation and the interview data revealed that 12 out of 15 documents were produced by the quality assurance department.

Participants' interview data also verified that the prescribed RM process was mainly used as a capability displaying or certification compliance tool and was not followed within the organization. The findings from analysis of the prescribed process, actual RM practices and the interview data were synthesized into the 'model in practice' presented in Section 5.6 for further analysis.

5.6 WIPS Requirements and Change Management Process Model in Practice

The model in practice was empirically constructed from the data collected from the participant interviews, analysis of RCM process, practices, artefacts and the observations made during the study of WIPS requirements change management process. The model in Figure 5.7 presents requirements and change related activities and their sequence that is closer to the actual RM practices in the organization. As discussed in Chapter 4, Section 4.6, the lifecycle activities covered in this model are divided into two main activities a) RE activities suggested by Kotonya and Somerville (1999) and b) Implementation and Testing activities. The lifecycle activities covered in this model are: 1) Elicitation 2) Analysis and Negotiation 3) Model/Design and Specification and 4) Validation

Negotiation and Prioritization. These activities are followed by two support activities 5) Implementation 6) and Test & Fix Cycle. Furthermore, these lifecycle activities were supported by configuration management traceability activities.

As mentioned in Chapter 4, Section 4.6., this research considers that when a requirement (or a change) covers all of the above mentioned lifecycle activities stated above, that constitutes its complete lifecycle. Unlike the prescribed model shown previously (Figure 5. 6) the model in practice (Figure 5. 7) does not separate requirements elicitation from change elicitation because during the initial phases both follow the same lifecycle activities. For the WIPS project, requirements change lifecycle activities and the associated challenges in managing these activities are discussed in Section 5.7 using the model in practice shown in Figure 5. 7.

The inputs for this process were (an RFP based) contract, data from existing system, user requirements and other project related material (GSD Inc., 2012g). Since the methodology adopted for the development of WIPS program was a mix of waterfall and prototyping the model in (Figure 5. 7) appears to be sequential. For requirements elicitation, analysis and initial design (prototyping) activities, a period of six months was stipulated in the contract before requirements could be signed off. However, since the development methodology also exhibited some properties of Feature Driven Development (FDD), batches of features from the design documents were formalized into detailed specifications before being developed incrementally. The batches of requirements and changes were implemented and went through test –bug fix cycles (internal, Alpha and Beta) before being finally released to production on a public website.

The model in practice (Figure 5. 7) also highlights both the formal and informal management of requirements changes enclosed in two clouds. This model depicts informal requirements change management activities within the two clouded regions. On the right hand side of the main lifecycle activities, informal change activities are depicted (in the clouded region). These activities take place prior to the design specification document signoff. If the change is accommodated by the vendor it returns to the requirements lifecycle activities circle otherwise it undergoes a negotiation cycle before being reprioritized or deferred by agreement. The left hand side of the main lifecycle activities again show the activities for accommodating change as previously noted (in the clouded region). Since many contractual requirements upon elaboration became changes, they passed through the informal activities' cloud (top right of Figure 5. 7).

Informal requirements management refers to requirement changes that are a) accommodated by the vendor without charging the client or b) done through the reduction of the (perceived) project scope by mutual agreement of the client and the vendor. Section 5.7.8.2 through to Section 5.7.8.4 discuss the formal and informal requirements change management in more detail.

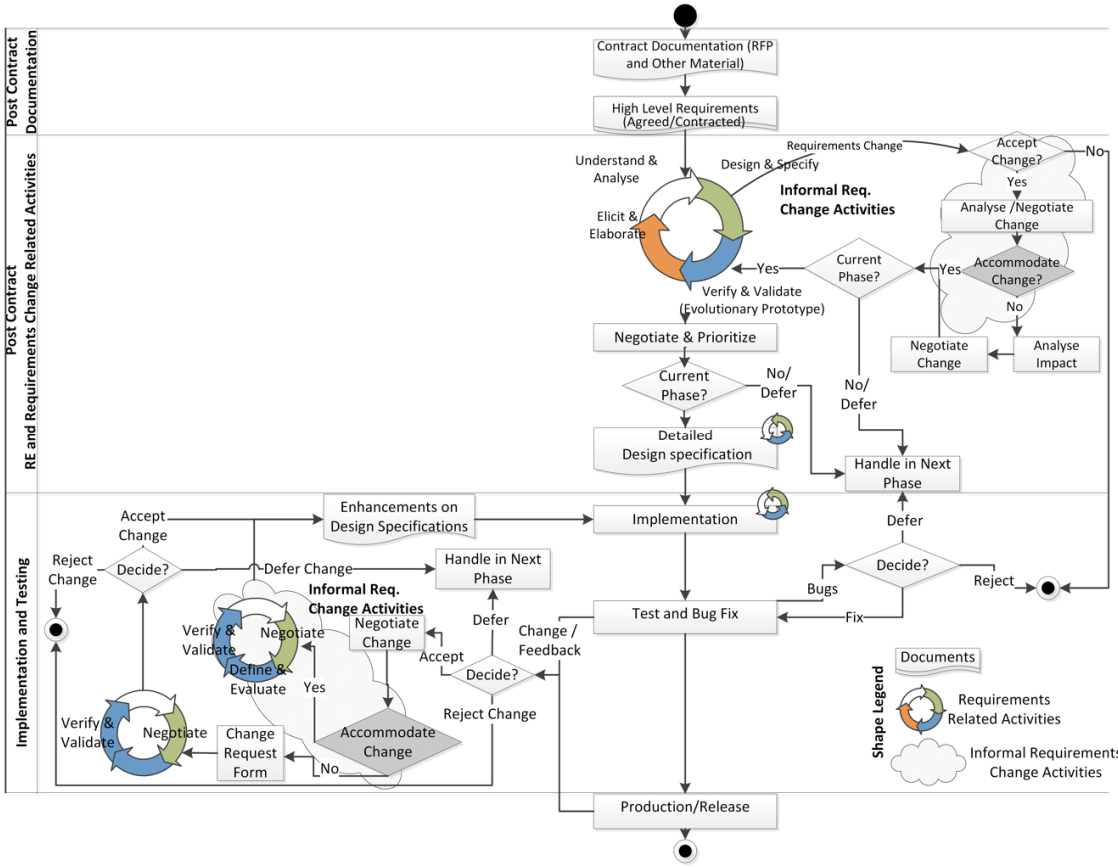

Figure 5. 7 Change Management Model in Practice

5.7 Analysis of the WIPS Requirements and Change Management Model in Practice

Analysis of the WIPS requirements management model in practice is based on the lifecycle activities of a requirements change as discussed in Section 5.6. For WIPS project changes in requirements started coming through after the contract was signed off at the beginning of the project. Therefore, in order to capture the complete lifecycle of a change the activities the model in practice starts from requirements elicitation till the implementation, testing and deployment. The challenges faced by the practitioners in carrying out lifecycle activities of a requirements or change are discussed and analysed in the following sub sections.

5.7.1 LCA1: Requirements Change Elicitation

Elicitation was the first activity in the lifecycle activities of a requirement or change. Although a set of high level requirements was agreed on and recorded in the contract, elaborations and confirmations of these requirements were required to develop a shared understanding of the contracted requirements. The following statements from the contract (FBO.Gov, 2011) highlight the (high level) nature of requirements for WIPS design and specification to be carried out by the vendor.

"The [WIPS Program] shall at least encompass the functionalities available at the [example product website] [link]. In addition, other [existing products] such as [product 1, product 2, product 3, product 4], and others as identified by the COTR, shall be incorporated into the [WIPS program] 1.0".

The statements above clearly suggest the need for elicitation and elaborations and clarification of requirements before design could be finalized for approval by the client Contracting Officer's Technical Representative (FBO.Gov, 2011) before it could be implemented.

Once a common understanding was reached on the initial project requirements and scope a list of requirements (or initial design document as was the case in the WIPS project) was shared with the client seeking further elaborations, clarifications and understanding. These elaborations revealed some issues with the requirements understanding at the very first instance during stakeholder interactions such as misinterpretations, problems identified in high level specifications, misrepresentations and misunderstanding. The challenges faced during elicitation activities are discussed in the following subsections.

5.7.2 Challenges with Requirements and Change Elicitation

The main challenges during the elicitation phase were the dependency of the offshore development team on the proxy client for elicitation and understanding of both requirements and changes, lack of domain knowledge of the development team, misalignment of understanding between the actual and proxy client regarding requirements. The dependency sometimes led to misinterpretation of the actual client's need in the prototypes developed causing delay and rework discussed in the coming sections.

5.7.2.1 Dependency of Offshore Team on the Proxy Client

The offshore team members were dependent on the proxy client/remote BA to transfer the initially elicited requirements and project knowledge obtained first hand from the

contract documentation and through his interactions with the client (FBO.Gov, 2011a). The vendor's remote BA was onshore with the client and interacted with them for requirements and acted as the proxy client for the offshore development team in Pakistan.

FRPK-WAA-03-2013: "Our first client is [the Proxy Client in US]… requirements come from there… he is a business analyst, he does the field work [gathering initial requirements from the client], does demos and provides feedback to us… [the proxy client] tells us what the project is… we understand requirements [from him]."

The high involvement and dependency on the onshore BA during the initial phases of requirements development and design discussed by the development team members is stated below.

FRPK-ZAE-04-2013: In [WIPS project's] case we had to fetch quite a lot of knowledge from him, but in [other two projects] it was not. In [WIPS project's] case we are really dependent on him"

FRPK-WAA-03-2013: So basically he is a business analyst as the [project] understanding is gathered by him in detail, what to do with the system, [then] he makes us understand in detail what exactly has to be developed… [the proxy client's] understanding [of requirements] is quite good".

The development team members did engage with the actual client occasionally but only to discuss major problems and to resolve major issues in requirements understanding. Even when the development team engaged with the client directly through teleconferences a follow up conference call was held 'explaining' what had happened and what to do next by the proxy client.

FRPK-ZAE-04-2013: "Right after every conference call or every webinar or something we sit together, and [the proxy client] says, ok this has happened and now we have to do this"

The development team had to rely on the remote BA's knowledge for developing design documents and other artefacts. Therefore, requirements related dependency was most high during the elicitation phase and remained high until the design phase as per one of the developers.

FRPK-MAI-03-2013: "when we [the development team] are in the requirements design or prototype development phase the involvement [from the proxy client/ BA] is very high … that [level of] dependency remains until the requirements design phase"

This dependency on the proxy client's knowledge meant that the development team had to verify and align their understanding with the proxy client and satisfy him for the accuracy of the design documents.

FRPK-WAA-03-2013: "we understand requirements [from the proxy client] and make design docs … we design screens in MS Visio… then we send it to the [proxy client, who] corrects them in two or three iterations and when it gets finalized [by the proxy client]".

During this phase the development team had to face a three tier approval process 1) local site approval (from team leads or managers), 2) proxy client's approval and 3) actual client's approval. This meant that if there was a gap in the understanding of requirements by the proxy client it would also get reflected in the development causing further effort in requirements communication, specification, design or fixing errors.

5.7.2.2 Lack of Domain Knowledge by the Development Team

The development team members did not have domain knowledge for the WIPS project and required assistance from domain experts to simplify technical requirements for their understanding. This deficiency was partly covered by the proxy client who himself was a domain expert but the client also hired a partner consultancy organization in the USA to help explain technical and domain specific issues to the development team. The consultants translated technical user requirements in an understandable way for the development team.

FRPK-DUI-05-2014 "[technical] requirements sessions were very technical and would simply go 'over our heads'…we would understand things later through [meetings with] consultant …[who were] there to help us understand the technical requirements in a plain English."

The development team worked with the consultants in order to verify their understanding of the technical requirements as well as their implementation in WIPS program which again proved an added layer in the development and approval process (See Figure 5.3).

5.7.2.3 Misalignment of Understanding between the Proxy and the Actual Client Requirements

During elicitation, the real challenge of requirements understanding came when the development team provided a demonstration to the client using a prototype. Although the prototype had already been approved by the proxy client however upon validation the client highlighted many issues based on requirements misunderstanding that required many changes. Overwhelmed with the changes and issues identified; the development

team was disheartened. They could see that their understanding of the requirements was not really aligned with the (actual) client's needs. The development team members had to be motivated again to modify the prototype for later validation to meet the client's requirements.

FRPK-DUI-05-2014: "We developed the prototype to understand the actual requirements…our team members were quite disheartened that the client basically disapproved and dismantled what we had developed as a prototype …So requirements were almost the same, but their interpretations drastically changed after those initial prototypes and demos."

The development manager considered it a positive activity but clearly the misalignment of requirements understanding was exposed by the activity and the development team had to do considerable rework on the prototype before it could become aligned with the actual client needs.

More importantly according to the company's policy the vendor had to accommodate (informally manage without charging the client) any changes in the agreed requirements due to vendor's misinterpretations of the requirements. The identified issues therefore had to be managed without invoking change management process by the vendor's development team and without any cost to the client. This essentially meant that the rework performed by the client team was not considered added work and they were not given any compensation for it.

Accommodation of these informal changes into the existing scope and budget of the project and the associated challenges are discussed in detail later in section 5.7.8.2.

Table 5.2 summarizes the challenges faced during elicitation activities.

LCA 1 Requirements Elicitation -- Challenges	**Challenge Description**
Dependency of Offshore Team on the Proxy Client	Overdependence on the proxy client by the development team for project knowledge, client interaction, domain expertise, requirements understanding provided the proxy client with greater authority. It also added a new layer for requirements verification and approvals that had to go through him which caused misalignment in requirements understanding and delays due to geographical and temporal distances.
Lack of Domain Knowledge by the Development Team	Lack of domain knowledge in the development team also added a new layer of requirements related approvals through domain experts. It added delays and cost to the project.
Misalignment of Understanding between the Proxy and the Actual Client Requirements	Misalignment of actual client's requirement due to misinterpretation by the proxy client caused lack of shared understanding of the actual needs. As it was seen as

	misinterpretation by the vendor BA, For the development team it meant extra work without any compensation.

Table 5. 2 Requirements Change Elicitation Challenges

5.7.3 LCA2: Requirements Analysis and Design

The initially elicited requirements were first analysed on the basis of the analysis design documents and GUI mock-ups which were created. After the initial approval cycles; first from the local development managers and then from the remote proxy client, these artefacts were shared with the actual client for their feedback to verify and achieve a common understanding. Based on the client's feedback, design documents and GUI mock ups were modified and again shared with the relevant stakeholders for further review and feedback or approval. The vendor team considered this as their analysis and design phase because design is done in parallel with the analysis.

The purpose of the analysis and design phase was to further refine understanding of the requirements through design documents and client feedback. One of the development managers and a developer explained the purpose as below

FRPK-WAA-03-2013: "We call it requirements analysis & design phase, it is requirements analysis but design is done with it."

FRPK-DUI-04-2013: "we take design phase as a tool for developing understanding on the requirements and we take active feedback from the clients…the main motive of our initial mock ups is to have a better understanding to meet the client's expectation. The stakeholders get a better understanding of their own requirements, after they see the prototype in somewhat working condition…based on the feedback we rebuild the part or [change] the design docs."

5.7.4 Challenges with Analysis and Design

Challenges during the analysis and design phase of this project related to creation and maintenance of a high volume of documentation, incorporation of continuous changes based on the client (both proxy and the actual client) feedback, having to satisfy virtually two clients (proxy client and the actual client), dealing with high level of involvement and demands of the proxy client, and calculating accurate estimation for the design phase. The detail of these challenges is discussed in the following sections.

5.7.4.1 Creating and Managing High Volume of Design Documentation

WIPS was estimated to be a *'large project'* as per the vendor's project size calculations (GSD Inc., 2012e) and scored a maximum of 500 weight points available in the calculation matrix. The criteria were based on project scope (number of interfaces,

number of installations, required documentation and number of resources) and time required (in months). The vendor team furnished a 165 page long comprehensive review report (GSD Inc., 2011) based on the project related literature (manuals, guides, reports), existing products (highlight their strengths and limitations) and requirements collected from the technical staff and end users. The review report had at least 11 documented versions and it served as a base document for the 75-page long Work Plan design document (GSD Inc., 2012c) for WIPS. Both the design documents and review report were provided to the client. The analysis also revealed that the work plan design document itself had had 10 versions.

For detailed design the work plan design document was further divided into features or modules and further design documents were generated. The document review carried out for WIPS project revealed that from the high level work plan design document (GSD Inc., 2012c) thirty-eight (38) individual design documents had been created as of 15 March 2013 according to the *list of design documents* artefact (GSD Inc., 2013b). Some of these individual design documents were found to have multiple versions for example Administration Site Design (GSD Inc., 2013a) had seven versions. One of the software engineers involved in design documentation talked about the complexity of the project and the features or modules involved.

FRPK-IME-04-2013: "[WIPS project] has got so many features and the design documents are almost 20 to 25 and many of them have been completed, I guess 20 have been completed and 6 remaining. Some are related to configuration and some are related to features."

Apart from the documentation exchanged and maintained with the client the vendor organization also had to maintain CMMI level II related process documentation to fulfil CMMI certification audit requirements. Since the WIPS project already had significant documentation produced it was also used as an exemplary project for CMMI level certification audit which meant that even more project documentation was to be produced and maintained (GSD Inc., 2012f).

The participants frequently reported challenges related to developing and maintaining heavy project documentation and change related artefacts in their interviews. This caused additional delay in creation, verification and maintenance of this documentation due to geographical and temporal distance and inadequate communication.

5.7.4.2 Challenges due to Lack of Updated Documentation

Faced with these challenges the development team modified their practices for managing project requirements. For example, in some cases not even creating a traceability matrix, avoiding to update change requests in the design documents, documenting only essential changes in the design documents, or making changes directly into the prototypes. The examples or such problems are presented here to illustrate these issues.

FRPK-DUA-11-2013: "We do not or cannot develop or maintain a traceability matrix because we get bombarded with many requirement changes so what we are trying to do is to write as many of them as possible within the design documents … there are many problems that follow this sort of practice."

The problem with this change in practice was that the use cases related to these changed requirements do not get updated and hence their test cases are not updated. Testing performed on the basis of the unchanged and older test cases resulted in confusion as the testing team classified changes as bugs. Furthermore, in the absence of test cases, the development team had to verify the implemented change on the basis of their understanding of the design documents. In many cases the problem arose when the developer's understanding of the design document did not align with the tester's understanding of the requirement or change and that caused conflicts between the team members.

FRPK-DUA-11-2013: "when receive requirements change requests and do not create or update use cases it creates the problem for the developer who sometimes fails to understand change … while testing when the testers look at the module or feature (with the implemented change) they get puzzled and worried to see that the developer has totally misunderstood the requirement and has implemented it incorrectly".

5.7.4.3 Serving Two Masters: Satisfying the Proxy and the Real Client

Requirements analysis and design was performed in consultation with the BA onshore with the client who also acted as the proxy client for the offshore development team. The proxy client's demanding nature and involvement in the design process was considered an additional challenge. Since, the BA / proxy client (onshore with the actual customer) was officially treated as the first client by the offshore development team and they had to incorporate all the changes suggested by him. Some of the team members hesitantly complained about the 'pixel level' involvement by the proxy client and having to make

precise and detailed adjustment in the artefacts prior to the proxy client's approval or satisfaction.

FRPK-WAA-03-2013: "[the proxy client] is involved at the absolutely 'pixel level' even on a single pixel that how it should be".

FRPK-ZAE-04-2013: he is even looking at things like 'pixel perfection'. He will go himself (inside the pages/code) and say adjust the alignment here, or correct that, he gets involved too much.

When the artefacts resulting from the analysis and design were verified and approved by the proxy client only then they were shared with the actual client for their feedback. This became a double verification and validation feedback process. One of senior software developers shared the experience of sharing prototypes with the client and the known practice of having to change the documents and GUI based on the client.

FRPK-DUI-04-2013: "[The client] feedback is adjusted in the design documents, and updated design documents are sent back to the client so we have all the track of initial design documents and then their revised versions based on those requirement changes or better understanding of requirements"

5.7.4.4 Poor Estimation for Design Effort

Effort estimation was reported by participants from GSD Inc. as the weakest area in their software development process. The organization had been facing estimation problems for a number of years. In case of WIPS project, estimation was even more problematic as there were no matrixes, repositories or measures available for the organization to use as a precedent.

FRPK-DUI-04-2013: "That is our weakest area we are still struggling with the measure and analysis things and since there are certain things applied on this project and we don't have anything to compare it to previous projects we are working on those areas we don't have good repositories for those measures"

Furthermore, not using a repository for planned versus actual effort particularly for design estimates resulted in a lack of insight on what went wrong and why. The organization therefore lacked the knowledge about any measures to rectify the poor estimation problems.

The development team was the main victim of poor estimation problems as they had to spend extra hours to make up for the estimation variances. One of the development

members verified the problem of under estimation and the consequences the development team has to face due to it as:

FRPK-WAA-03-2013: "It [poor estimation] is covered by working more hours and through late sittings [by the development team]."

Since the contract was time bound, poor estimation also placed pressure on the development team to achieve various milestones and produce deliverables within the given project deadline.

Table 5.3 summarizes the challenges faced during requirements analysis and design activities.

LCA 2 Requirements Analysis and Design-- Challenges	**Challenge Description**
Creating and Managing High Volume of Design Documentation	Creation and maintenance of high volume of design documentation was challenging. Having to rely on the domain expertise of external consultants and the proxy client both for artefact creation and verification in a GSD environment was very difficult for the offshore development team and caused delays to the project due to distance and inadequate communication.
Poor Estimation for Design Effort	Faced with the heavy workload of documentation the offshore development team began to skip developing or maintaining important requirement artefacts e.g. use cases and test cases for requirements changes as well as the traceability matrix. This had a ripple effect on the downstream activities where development and testing was affected due to lack of important requirements related artefacts.
Serving Two Masters: Satisfying the Proxy and the Real Client	The development team had to go through a double validation loop for their design efforts. The first verification and approval had to be done by the proxy client and the second by the actual client. This caused delay for the project and frustration for some of the team members.
Poor Estimation for Design Effort	Poor design effort estimates due to lack of adequate estimation techniques caused the development team to spend extra effort for which they were not compensated. It also put pressure on the development team which had to achieve various milestones and produce deliverables according to the project deadline.

Table 5. 3 Requirements Analysis and Design Challenges

5.7.5 LCA3 and LCA4: Requirements (and Change) Specification and Validation

Requirements for the WIPS project were specified initially in the contract awarded to the vendor. These RFP based high level requirements went through a six-month analysis and design phase and were specified with more details and then signed off by the client. The project did not have a separate phase for specification as the requirements agreed at the

end of analysis and design phase were considered as requirements and design specifications.

The development for the WIPS project was carried out with some resemblance of Feature Driven Development, batches of these design specifications (in textual format, mock ups or prototype) were implemented incrementally. As the project progressed and batches of design specifications were being implemented, changes in requirements started to emerge. Therefore, the agreed on design specifications went through further cycles of changes, clarifications and validation by the client during the course of the project. Client-suggested changes (with significant impact) in the design specification went through the in place WIPS change management process. The process activities for managing change involved: creation of change request form, change impact analysis, change control board review, change approvals (or rejections) and validation by the client.

FRPK-DUI-04-2013: "If the change comes after the design phase, then we send them a formal change request form where we capture their 'change' description and we put our analysis of impact and the extra effort required and the costing of that effort... they can see, each and every part of that CRF and send their feedback, approve or disapprove accordingly."

5.7.6 LCA3 and LCA 4: Challenges with Requirements (and Change) Specification and Validation

The following sub sections talk about the challenges in specifying changes in design specifications.

5.7.6.1 Specification Using Prototype Instead of Specification Document

Maintenance of change i.e. updating multiple change related artefacts required additional time and effort and was often skipped since the management was hard pressed for meeting the deadlines. The focus for the developers was also on the implementation of changes rather than updating documents. Although the team knew that keeping documentation updated was a good practice, however it was not carried out regularly due to the heavy documentation effort required.

FRPK-DUI-04-2013: "what I mean to say is that for a single change request you have to update many artefacts, which itself is a time [and effort] consuming process. At times, this is the reason that the process does not get followed completely. So what we do instead is that we focus on implementing those CRFs then and there and try to come back and revisit these documentation things later on. The problem is that it is not a very good practice."

The frequency of changes was quite high and due to time pressure the development team started to implement changes directly into the prototypes without making necessary changes in the design documents.

However, making changes on the prototype and not updating the documents created challenges for change awareness and verification and resulted in requiring frequent deployments because some of the changes made by developers were not integrated into the latest build.

FRPK-ZAE-09-2013: "reflection of changes in the prototype however has the downside of having to make frequent deployments…when we are about to give the demo sometimes we realize that the changes made by the other developer are not included in the demo version so at that time you have to do another deployment, so these back and forth deployments waste a lot of time. It is a big problem with making changes directly in the prototype."

The organization was actively searching for solutions that could resolve the problem of having to update so many documents.

FRPK-DUI-04-2013: "Could there be a better way that we don't have to update multiple artefacts for example ideally one if that is not possible it should not be more than two."

5.7.6.2 Inaccurate Effort Estimation and Change Impact Analysis

Changes in requirements were specified initially on Change Request Forms and then in the design specification documents. Calculating change impact in terms of effort required to implement change was a weak area for the organization. Poor estimation was a challenging area for the organization and in one of their previous projects the overall design effort was significantly underestimated (planned 169 vs. actual 460 hours. The WIPS project was no different and quite often the specified changes were under estimated. One of the developers explained the situation regarding the persistent estimation problem faced by GSD Inc. during estimation.

FRPK-ZAE-09-2013: "Impact analysis is our weak area… for example if there is a portion of impact which we did not analyse initially and it was a dark area, it later pops up and since we had not done its time estimation we face problems. Similarly, we often miss out the impact of integration when giving out estimates of development work. So if two people will implement a change and we estimate the development effort as X plus Y hours of the development time from the two developers, but we do not consider Z hours for integration time."

Missing out on accurately identifying impact points and effort on change request forms and change specifications resulted in additional burden on the team in order to fulfil initially provided estimates.

FRPK-ZAE-09-2013: "That [wrong estimate] pops up later as a deficiency of impact analysis, so the burden is then on the developers to come to the office even on weekends or do late sitting to solve it. We did face that problem throughout the analysis and design part of the changes."

Being poor at estimation and having to adhere to the estimated hours initially proposed (mainly because of a governmental client) the development team had to put in extra effort

FRPK-WAA-08-2013 "In this project we have spent a lot more time than what we had initially estimated"

The estimated effort hours were clearly different from the actual number of hours and resulting in missing the project scheduled deadlines. Therefore, the team members had to put in not only extra hours but extra days as well.

FRPK-DUI-04-2013: "Yes we had to work so many hours last week as we had a demo according to our scheduled which we have to complete… due to some extra work we had already scheduled our Saturday as a regular working day. As per our schedule we were running behind."

Similarly, answering the question about who bore the burden of the wrong estimates the software engineer replied

FRPK-WAA-08-2013 "We [the development] team."

5.7.6.3 Powerful Proxy Client and Problems with Effort Negotiating

Accurate impact analysis and change effort estimation was additionally challenging due to the role of the powerful proxy client. Effort estimates shared by the development team were often unduly cut down by the proxy client. Since the proxy client was also the CEO of the company it was difficult for the team members to fairly negotiate the proposed effort estimates.

A software developer shared the general experience of negotiating the effort hour and reported that the developers had to really suffer to finish the work when the number of hours were reduced by the proxy client.

FRPK-MAI-03-2013: 1st interview "we generate the CRF and calculate the impact and send the CRF to the [the proxy client] … [who] asks us to cut it down because the company has to retain the clients and do not want to scare them off…[but] our loss is that the developer suffers, he has

to put in his weekends and has to do late sittings…he has to work in a stretched mode….it is stressful."

A software engineer shared his experience of negotiation with the proxy client and the fact that the developers have to put in extra hours to finish the work.

FRPK-WAA-03-2013: "before the CRF goes to the actual client from the company our [proxy client] cuts it down quite a bit … when it comes to us forcefully asking to just 'do this work' then you are spending extra hours or doing it [from home] sitting in the bed".

Sometimes when the proxy client insisted on reducing effort hours the development manager, unable to refuse, had to entertain the request by proposing to assign the task to a different developer who could finish the job a bit early.

FRPK-OOI-04-2013: "Sometimes we straight away tell [the proxy client] that we would require exactly 20 … this work cannot be done in 10 hours at any cost… sometimes we will just say ok, we will assign resource A rather than resource B, because resource A is quicker in development than B so it will be equal to some sort of 14 hours or 15 hours …we sometime negotiate in this manner."

Similarly, another development lead reported that in order to speed up delivery to meet an internally (or externally) imposed deadline the developers either put extra hours or add another developer to finish the task quickly.

FRPK-IME-04-2013: 1st interview "if the work is of 20 hours, we tell [the proxy client] that it is going to take these number of hours and it cannot get reduced, but in order to meet the deadline we put two resources instead of one and put in extra hours, instead of working 8 hours we will put in 10 -12 hours a day so we fulfil it like that."

The development team however did waste time in carrying out these negotiations and often had to agree to the proxy client to reduce the number of hours because according to him the actual client might not agree to the development team's estimates.

FRPK-OOI-04-2013: "the US office [the proxy client] quite often negotiates the estimates we waste a lot of time…, they tell us you have to reduce the number of hours here because, the client will never agree on that [estimated] time."

However, one of the development managers said that the team is allowed to work on their proposed effort hour estimates. He suggested that the developers are not forced to reduce their estimates and if the team members followed their schedule there would be no need for spending extra effort by the developers or late sitting.

FRPK-DUI-04-2013: "Whatever the time is decided by our team that much time is given to them, we don't reduce their time…. I believe if the schedule that we created in consultation with our team if that is actually followed by each and every member of the team then we won't be having the requirement of putting in any extra effort or late sitting."

5.7.6.4 Delay in Feedback from the client

The development team faced the problem of delay in requirements and change validation as well as getting feedback from the remote client. The nature of the project which involved a governmental client was reported as one of the causes for the extra delay in the feedback. The delay meant that the development on that particular requirement had to wait until a response was received. He considered that getting approvals from the relevant stakeholders in a bureaucratic model can take even longer causing further delays in change implementation as experienced in WIPS project.

FRPK-DUA-11-2013: "when we send the CRF to the client we stop the development work on that requirement and wait for the client's approval or feedback. …we are working for a governmental client …since bureaucracy is involved so everything gets done slowly…generally the client takes too long and sits with the CRF sometime for a month."

Table 5.4 summarizes the challenges faced during requirements and change specification as well as validation activities.

LCA 3 and 4: Requirements (and Change) Specification and Validation--Challenges	**Challenge Description**
Specification Using Prototype Instead of Specification Document	Time pressure and heavy documentation made the development team specify changes directly into the prototypes without making necessary changes in the design documents. It created challenges for change awareness, verification and testing that resulted in faulty deployments as some of the changes made by developers were not integrated in the latest build.
Inaccurate Effort Estimation and Change Impact Analysis	Missing out on accurately identifying impact points and effort on change request forms and change specifications resulted in additional burden on the team in order to fulfil initially provided estimates.
Powerful Proxy Client and Problems with Negotiating Effort Estimates	The powerful proxy client quite often asked the team members to reduce the reported effort hours. The developers, as a result, had to do extra development or design effort due to meet unrealistic estimates promised to the proxy client. This was stressful for the developers and caused a loss of significant project time on a regular basis due to these negotiations.
Delay in Feedback from the client	Governmental client and geographical distance was causing undue delay in the feedback from the client. This

	was a challenge for the project time line and on resource allocation for the vendor team.

Table 5. 4 Requirements (and Change) Specification and Validation-- Challenges

5.7.7 LCA5: Requirements Implementation

Requirements implementation started once the design specifications were agreed on and the main design document was signed off. As noted in section 5.7.6 the development was carried out with a slight semblance to the feature driven development methodology. Implementation and validation of requirements therefore were carried out in iterations and demonstrated in multiple releases (e.g. alpha, beta, pre-launch and launch).

FRPK-DUI-04-2013: "once the design documents are signed off from the client and the client and our team are on the same page for requirements understanding, acceptance of the UI and workflow of the application, if all of that is conveyed [and agreed] to by both the parties then we start development."

These iterative cycles of implementation and validation resulted in identification of new requirements or modifications to the existing ones with a possible impact on the project scope. Among these identified changes some were categorized as out of scope and were treated informally, while others (having a 'tolerable impact') were included in the scope by the vendor and were treated as part of scope and implemented without any cost to the client.

5.7.8 LCA5: Challenges with Requirements Implementation

The following sub sections deal with the challenges identified with requirements implementation activities. The challenges reported here are mainly reported by the vendor organization. The client perspective is partially covered by the inclusion of proxy client's views.

5.7.8.1 Continuously Evolving Requirements and Challenges with Baselining

Both the development team and the client's understanding of the requirements improved over time and resulted in transformation of the initially agreed on scope and proposed solution. According to the team lead providing the client with multiple prototypes and releases opened the door for changes in requirements to pour in. However, it was reported that prototyping bridged the gap of understanding of the requirements and covered up mistakes in the development team's understanding of requirements.

FRPK-ZAE-04-2013: "The project understanding was evolving rapidly and the changes were being reflected mainly in the prototype rather than in the design documents…by the time we

reached the alpha release our initial understanding was totally transformed from what we saw as the solution in the beginning."

However, with the requirements changing so rapidly, members of the offshore team (especially in the QA department) faced the problem of base lining requirements. According to the quality assurance manager the changes kept coming in, even after the beta release was launched.

FRPK-DUA-11-2013: "the requirements are changing so frequently for some projects that it's not possible to baseline them at any time. In some cases, the changes come after the beta release…. so we don't know when to baseline."

With the huge influx of changes in requirements the required implementation effort increased significantly. Since the contract was time bound the additional implementation effort had to be covered within the same timeline. This proved quite challenging for the development team which not only had to spend extra hours but also had to sacrifice some of their well-deserved weekends.

5.7.8.2 Accommodation of Requirements Changes Informally

Accommodating changes informally i.e. implementing change requests without invoking the RCM process and without charging any cost to the client, was a challenge for the development team. It kept adding to the required effort estimated for the project. One of the problems was the management philosophy, as it did not consider modifications to requirements during analysis and design phase as changes.

FRPK-DUI-04-2013: "changes [that] come during the requirements or design phase (which we consider a tool for developing understanding on the requirements)… we do not consider them requirement changes we rather consider them as 'better understanding' of the same requirements."

If the management considered that the scope of the change was 'insignificant' management (proxy client & managers) the development team had to informally accommodate it.

FRPK-DUI-04-2013: "if both the parties see that there is a small change [in scope] and it is absorbable by the initially proposed cost to the client then we do that on our end."

However, the offshore development team members had to put in extra effort to implement such 'insignificant' changes. The offshore developed team was also faced with an implicit internal threshold of when to use a CRF and development team members were

'encouraged' to simply carryout 'small' development tasks that involved a development effort of less than five hours.

FRPK-ZAE-04-2013: "See first it there has to be a consensus from our CCB members, to see if it really is a change [worth going through the process of CRF] or not, if it is a change of four hours, then we don't make a CRF of it at all."

If the change demands a significant effort based on an informal evaluation and negotiation (not involving CCB and CRF) then the change is deferred or the vendor demands additional cost [not the case in this project].

FRPK-DUI-04-2013: "If we see that it [a change request] is going outrageous and we need much more effort and resources then we convey it back to the customer."

The problem with this approach was that the decisions related to accommodating and entertaining such (informal) changes in requirements were taken by the higher management. This meant that until the management thought the effort is getting "outrageous" the development team kept on spending more and more hours on it.

5.7.8.3 Influx of Informal Change Requests from the Proxy Client

The powerful role of the proxy client meant that he could use his authority to mandate changes in requirements without having to justify them. When considered individually the changes often appeared insignificant, according to one of the quality assurance manager, however taken as a whole they required very significant effort from the development team. The high frequency of these changes was even a bigger challenge for the team because it not only needed additional effort it also required significant testing to ensure the desired quality.

FRPK-DUA-11-2013: "The customer has simply given us the data that we need to present in a useful, appealing and better way. … the proxy client says that we would build graphs to present the data, and the next day we are told that we want to particularly make a certain type of graph and the team agrees but the following day the he (proxy client) would say no no I want you to develop a 3D graph. So almost every second or third day the requirements keep changing. But the proxy client does not even consider those as requirements change, he considers that because we have to ultimately show the data so it doesn't matter if we make changes in the way it should be presented while learning at the same time to see how it would look best."

These continued accommodations of requirements changes did add up to possible delays in the delivery which had to be compensated by developers putting in additional time to stay on schedule.

FRPK-IME-04-2013: "Yes we do have to put in the [extra] time. There are so many factors kicking in, it really is impacting our schedule."

The informally accommodated changes were not directly reported as a challenge in the project by the participants but did report that they were spending time on carrying out these 'small' and 'insignificant' changes. One of the managers from the development team agreed that accommodation of informal changes did put extra burden on the development team but only during the later phases of development.

FRPK-DUI-04-2013: "yes they [informal accommodations] put pressure on the team when the change is at the very tail end of the project but when the project is in the beginning or in the design phase I don't think so"

5.7.8.4 Resource Allocation to Implement Formal Changes in Requirements

Allocating the tasks to implement change to the right resources was challenging for the project managers. Coordinating their availability and time and trying to assign change related tasks to the developer who had already worked on the change was found quite challenging.

FRPK-WAA-03-2013: "In the [finalized] CRF the estimated date for requirements [change] implementation is given and the completion date so then have to consider the resource availability, you look at the module to be impacted identify the person who has developed it… you normally try [to engage] the same guy who was involved in it… you check his availability and then you have to manage it accordingly."

The management had to avoid allocating developers who were working on other projects to be assigned on a particular change even if it was urgent. The management considered that the other projects team members and their clients did not like resources with key project and domain knowledge to be switched between projects. So even if the developers were not doing work on a particular project they were given some relief time according to the one of the project managers. Of course this meant that the project requiring immediate support from the additional team members had to wait adding delay to the planned schedule.

FRPK-DUI-04-2013: "the psyche of our team members is that the developers they don't like being moved around and assigned to different projects they are like 'why do you keep rolling me between projects' so they get frustrated so at times you give them some spare time so that they feel easy."

Table 5.4 summarizes the challenges faced during requirements and change implementation activities.

LCA 5: Requirements (and Change) Implementation--Challenges	Challenge Description
Continuously Evolving Requirements	Requirements continued to evolve rapidly as the project progressed and the offshore team members struggled to maintain the baseline for requirements and manage changes.
Accommodation of Requirements Changes Informally	The challenges due to accommodating informal changes included working under pressure on the agreed on delivery date, putting in extra hours, late sitting, problems in base lining requirements and managing requirements.
Influx of Informal Change Requests From the Proxy Client	Having to spend extra effort and time to incorporate many informal changes (considered insignificant) and not being able to say 'no' to the proxy client was a challenge for the development team members.
Resource Allocation to Implement Formal Changes in Requirements	Managing resources in multiple teams and allocating members to different team without much disruption and concern to both the members and clients was a challenge
Common Challenges with Implementing Formal Changes in Requirements	Inaccurate impact analysis and effort estimation placed pressure on the development team to implement changes in an unrealistic time was a common challenge for LCA 3, 4 and 5. Managing and updating multiple change related artefacts was also a common challenge for LCA 3, 4 and 5. Similarly, estimating accurate effort required to implement the change and negotiating estimated effort hours with the proxy client was a common challenge. The implications of these challenges were having to spend extra effort not reflected on the estimates reported and the developers having to suffer as a result by putting in more work hours.

Table 5. 4 Requirements (and Change) Implementation Challenges

5.7.9 LCA 6: Testing and Deployment

According to the contract signed between the client and the vendor testing had to be performed in multiple releases for the WIPS project. Since the contract outlined broad level project objectives, deliverables and requirements, a change could come at any given point during the project. The timeline related to WIPS testing and feedback according to the contract is shown in Figure 5.8.

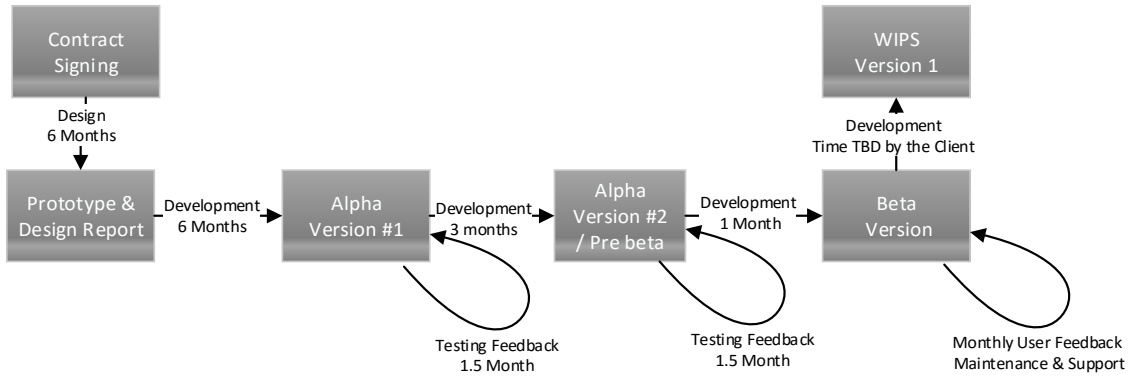

Figure 5.8 Testing and Feedback Cycles for the WIPS Project

Although not apparent in Figure 5.8, the development team also had to implement and test changes requested by the BA onshore with the client who was also acting as the proxy client. However, these changes were either implemented verified informally and without invoking the change management process or were deferred for later consideration.

According to the contract each release after Alpha Version #1 had to incorporate (or at least consider) the testing feedback/comments made by the client's external testing team. This implicit constraint can be ascertained by looking at some excerpts from the contract. For example, *"Alpha Version #2 shall be produced within 3 months from receiving the [the client's] comments on the Alpha Version #1, and made available to the [the client]."* Similarly, *"Beta version of the LTPP Web Interface Program 1.0 shall be produced within 1 month from receiving the [client's] comments on the Alpha version #2." "The contractor shall submit to the [the client's] User Comments and Feedback Summary Reports monthly to document the user feedbacks, plus resolutions, as appropriate.* Deciding the degree of appropriateness to include a '*comment'* in a release was a matter of negotiation between the client and the development team.

5.7.10 Challenges in Testing and Deployment

Challenges faced during testing and deployment activities are discussed in the following subsections.

5.7.10.1 Lack of Awareness about Formal and Informal Change Requests

The development team faced problems with the awareness of both formal and informal changes. According to the quality assurance (QA) manager, the QA department was not kept up to date with the change requests that came after the development started.

FRPK-DUA-11-2013: "The first and foremost problem we face regarding a change is that we don't know actually a change has arrived."

The QA department was often informed about the changes only after they were implemented and were ready for testing.

Similarly, informal change requests (made by the proxy client through emails or late night phone calls) were often discussed only with the offshore development manager and not with the QA department. Team members from the QA department were often unaware that an (informal) changes has arrived and even been implemented in a particular build or release. For the testing team it was difficult to identify changes from bugs as they did not know why a particular screen behaved differently.

FRPK-DUA-11-2013: "Due to the geographical distance meetings are held during the overlapping hours mostly late at night between the development team manager and the proxy client … they discuss and decide a change and the development manager writes it down on the paper or note pad…the day the developer is asked to implement it but the QA department does not even know that a change has occurred."

Since these changes were not part of the available design documents the QA team either entirely missed testing those changes or they reported them as bugs. The QA manager attributed this change to the geographical distance and suggesting it would not have been a problem if the teams were collocated.

5.7.10.2 Problems with Understanding and Testing Informal Change Requests

Change implementation without involving the QA department created problems of misunderstanding and confusion between the team members during testing. Informal change requests often did not have any documentation available. The testers often became aware of the change only after it was implemented and came for testing at which point they had to chase up the developers who implemented the change to be able to test it.

FRPK-DUA-11-2013: "The tester gets to know about the change only when he looks at the actual screen itself and goes like 'why is this screen appearing or behaving different?"

Such lack of awareness caused by miscommunication between the development manager and the QA manager led to conflicts between the QA and development teams. Lack of involvement of QA team in requirements change related discussions was not good for the moral of the team. According to the QA manager, the team members felt rejected because they were not made aware of changes and not called in to participate in change related discussions.

5.7.10.3 Inadequate Test Cases

Due to heavy influx of requirements change and the resulting load of additional documentation some essential testing related document (e.g. Use Cases and test cases) were either not created or not maintained by the QA department. At times only one test case was produced to test an entire webpage. This often led to the problems during integration testing where most of the changes were missed.

FRPK-DUA-11-2013: "Only one test case was developed to test the entire of the screen. There are many problems that follow this sort of practice that some test cases get missed out during the integration testing."

5.7.10.4 Challenges with Manual Code Deployment

In this project the development team was not using automated code deployment feature for their code management. So whenever requirements change requests (both formal through CRF and informal) were implemented, code deployment became a problem.

FRPK-ZAE-09-2013: "Another major weakness is that we do not do automated deployments. So when changes are being deployed some of the files are missed or lost."

The problem with faulty and incomplete deployments became more complicated when the deployed versions were tested. The testers identified and reported the missed functionality due to loss of code or a missing file as bugs. The builds or releases with missing files or code had to be re deployed and retested.

FRPK-ZAE-09-2013: "so it [missing code files] creates problems when testing because although they are changed but since they are not deployed they are picked as bugs. Finger pointing and blame games start when the developer says that the tester did not test it properly on the staging server and the QA says that the developer did not deploy it properly."

Manual deployments and lack of adequate code management resulted in issues popping up in client demonstrations. This was embarrassing for both the development as well as the testing team.

FRPK-ZAE-09-2013: "We often give demos on the staging server so when [the proxy client] is giving a demo and if we have not tested something properly on the staging server then bug pops up".

The development team had to consider tools to support automated deployment such as TFS that would manage the deployment issues and the code loss.

Table 5.5 summarizes change related challenges faced by the practitioners during testing activities in the WIPS project.

LCA 6: Testing Challenges--	**Challenge Description**
Lack of Awareness about Formal and Informal Change Requests	Testing prior to release or client demonstrations was challenging as the testers were not made aware of the changes in requirements. Changes in software behaviour was treated and reported as bugs and caused conflicts between the development and QA team members
Problems with Understanding and Testing Informal Change Requests	Testing informal requirements changes was additionally difficult because the quality assurance department did not know how to test something for which they do not have any information on. For understanding the change request they had to locate the person who had implemented it which was inconvenient, challenging and time consuming.

Lack of Involvement of QA Team for Change Related Activities	The quality assurance team felt rejected when they were not made part of requirements change related discussions. This brought their moral down and caused them to be less keen participate in project related activities.
Inadequate Test Cases	An inadequate number of test cases were being generated which resulted in lack of testing coverage for example during integration testing and regression testing.
Challenges with Manual Code Deployment	Not using code management software to do deployments was resulting in data being lost or not included in the build. This resulted in rebuilding and retesting effort but also caused embarrassment during client demonstrations.

Table 5. 5 Challenges with Testing

5.7.11 Support Activities Traceability and Configuration Management

In the WIPS project traceability and configuration management were not directly reported as major challenges by the participants. However according to the quality assurance manager the lack of traceability coverage and the practice of not updating or creating change related documents (such as design documents update, use cases and test case) resulted in problems of change understanding and later testing issues as covered in the previous section.

The next Section 5.8, talks about the GSD challenges of WIPS project due to its geographically distributed nature of software development and multiple sites and stakeholders involved.

5.8 GSD Challenges

The WIPS project was carried out in a geographically distributed development environment. The challenges identified based on the three distance dimension (geographical, temporal and socio cultural) are discussed in the following subsections.

5.8.1 Challenges due to Geographical Distance

Geographical distance was considered a big problem for requirements and change related communication and coordination activities. The development manager suggested distance to be the main problem rather than the time zone difference.

FRPK-DUI-04-2013: "I think the distance itself, it is not the time, the GSD, I think the biggest challenge in GSD is this distance."

5.8.1.1 Distance and Dependency on the Proxy Client for Decision Making

The development team heavily relied on the proxy client for his approvals and managerial decisions throughout the project. Distance between the proxy client and the development

team however created problems for communication, both face to face and informal. This lack of communication caused significant delay and challenges for the development team. The development team manager suggested that whenever the proxy client was onsite with the development team it speeded up project activities. He also added that physical presence of the proxy client with the offshore development team made issue resolution much faster.

FRPK-DUI-04-2013: "since we have to take so many approvals [requirements, design, development and general] from him [the proxy client] so whenever he is here [onsite], things speed up automatically".

The development manager added that the lack of proxy client's physical presence affected communication. The organization added the night shift to cover for the lack of overlapping hours and improve communication with the proxy client but still could not prevent communication gaps arising because of distance.

FRPK-DUI-04-2013: "Lack of face to face meetings and the absence of facial expressions add to communication problems, physical availability matters a lot. We even have a night shift, which has a full working hour overlap with the proxy client but they can only talk over the phone and through meetings. So that interaction is not close enough and is not as frequent it definitely brings in communication gaps."

The proxy client had to make occasional visits to the offshore development site to somewhat bridge the communication gap. It allowed frequent interactions and face to face communication among the team members. However, the prohibitive cost of travelling meant that these visits had to be planned carefully. These visits were planned at the start of a new project to develop initial requirement understanding and work on design documentation. These face to face interactions were considered quite helpful by the development team in transferring project knowledge enhancing project understanding of the offshore development team.

5.8.2 Time Zone Difference

Time zone difference was among the most commonly reported challenge by the participants. The development team members considered the time zone difference to be a barrier to collaboration which caused problems such as communication delays, stress, and suboptimal usage of resources. Although the project members did try to manage the time zone difference by working late hours and trying to coordinate work activities during overlapping hours, time differences remained a challenge.

Some of the important challenges and their implications are discussed in the following subsections.

5.8.2.1 Time Zone Difference and Communication Delay

One of the biggest problems faced by the development team members was the delay caused by the time difference in the WIPS project. One software engineer considered that the lack of overlapping hours and the dependency of development team on the proxy client for requirements related activities resulted in delay and frustrations.

FRPK-ZAE-04-2013: "Time difference is the biggest problem….. I would send an email, and wait [because of time difference] for him to get up and reply or call me…and there might be questions from the development manager or someone else and [the proxy client] has to answer all of those so my turn would come after a while."

There were high demands on the proxy client's time both by the offshore development team members and by the actual client that caused further delays in communication and feedback. At times the proxy client was either not available or even when he was available the offshore team members had to wait for their turn to get a response from him.

FRPK-WAA-03-2013: "We [the offshore team members] coordinate with only one person [the proxy client] in the USA…sometimes [the proxy client] is not even available [to answer our queries]."

However, according to the development manager the time zone difference became a problem only when there was a critical issue and the proxy client was unavailable.

FRPK-DUI-04-2013: "Time [difference] is not always a problem, it becomes a problem when we need a response from [the proxy client] on urgent basis and [the proxy client] is not available or cannot respond at that time."

5.8.2.2 Late Sittings due to Lack of Overlapping Hours

Lack of overlapping hours between the development and the client sites was reported as a problem because the development team was dependent on the proxy client's domain knowledge and expertise. One of the team leads reported:

FRPK-ZAE-04-2013: "In [WIPS Project] we have to get project knowledge from [the proxy client] which is quite a challenge with lack of overlapping hours…when there is this kind of dependency then time difference becomes a problem."

The development team had to be both flexible in their working schedule and do late sittings to extend the overlapping hours with the proxy client. One of the software developers considered time difference to be an issue only with the proxy client and not with the actual client. Some of the developers in the morning shift were facing the

problem of working extra hours in order to bridge the gap between both the afternoon shift and the proxy client's time.

FRPK-MAI-03-2013: "Our [proxy client] is in USA… at 6:00 pm our time [the USA branch's] working time starts and the developer who comes at 9:00 o clock in the morning and his day ends at 6:00 pm which is the time US office hours starts so you have to stay for the meeting with [the proxy client] and sometimes stay even longer to complete work assigned late to you."

One of the female team leads related the difference of time directly with the problem of late sittings and having to put in extra hours. This was additionally challenging for females because they belonged to a particular culture where it is not considered appropriate for females to come home late at night.

FRPK-IME-04-2013: "Difference in time leads to the need of late sitting."

The problem of late sittings was intensified when a milestone or a deadline approached. One of the software developers shared his views on the frequency of late sittings and the reasons.

FRPK-MAI-03-2013: "it [late sittings] get frequent when for example when your iteration is close to getting completed then even your nights or on."

Since there were many deliverables and milestones mentioned in the project contract even the irregular late sittings became stressful. Even with the facility of having flexible working hours, late sittings started affecting personal life of the developers.

FRPK-ZAE-04-2013: "The time problem should get resolved… our daily life routines are getting disturbed. [the development manager] comes around 12:00 pm and I come around the same time and we are going home at 9, 10 pm or later and our personal life gets disturbed."

5.8.2.3 Time Difference as the Cause of Inefficiency and Stress

According to one of the developers the difference in time both with the proxy client and within the offshore development office was causing team related problems and was leading to underutilization of resources.

FRPK-MAI-03-2013: "The major problem is time [difference], because of the difference in working time between the management and the developer work efficiency and resource utilization are not optimum …time difference is causes clashes between the developer and the management and the developers are going under stress."

Once the status reporting started between the project management team and the proxy client certain development requests would come which the developers were asked to

finish before they could go home even when they had finished their regular working hours.

FRPK-MAI-03-2013: "when the proxy client asks about the progress and the [offshore team] management gives the status [during the overlap time] … at that very time [the proxy client] starts asking "ok we need this and this right now" and the [offshore office] management tells the developer that they have to finish it before they can go."

He continued to suggest that the pressure of deadlines and the frequent development related requests by the proxy client to be finished on the same day even when the developers have already finished their regular work hours directly led to work stress for the developers.

FRPK-MAI-03-2013: "when these [late requests] come an d you already have a lot of bottle necks or the deadlines are looming and you have to give out the deliverables, then you have to run in a stressed mode."

Communication with the proxy client were also stressful and challenging especially in pressure situations where deadlines were approaching which had its effect on participants' moods. During these demanding times several communication and collaboration problems were reported to have sprung up.

FRPK-WAA-03-2013: "the problem is that he [the proxy client] is one person who has the pressure of done many things [based on his many roles] … most of the times he gets quite annoying when pressure increases."

In response to a question about recording proxy client meetings for future references the development manager casually shared that those meetings are not pleasant to record.

FRPK-DUI-04-2013: "the way, those meetings [with the proxy client] are not that pleasant or that professional to be recorded".

The proxy client (also the CEO of the company) did not see time difference as a problem, rather the 12-hour difference was considered an advantage by him:

FRPK-IAE-04-2013: "Time difference is not a problem because we have a process in place. We have 24-hour support for our client because we are running three shifts [morning, evening and night]. We have a ticketing system which always available to the client. …for me time zone difference is actually an advantage. It works out to be an advantage rather than a disadvantage."

5.8.3 Language Difference

The difference in language (English vs. Urdu) between the client and the development team members was considered a problem by four out of seven participants. Difference in language was one of the main challenges in requirements or change understanding and communication for the offshore development team. The project management team was aware of the limited language abilities of the development team and their lack of confidence in speaking English. Therefore, in an effort to keep a good image of the company in client communications only the managers and team leads with good (English) language skills were allowed to talk.

5.8.3.1 Problem in Understanding Requirements

One of the team leads who had worked with many teams reported that the problem in language was not confined to any particular team rather it was faced by almost all the teams.

FRPK-IME-04-2013: "I have worked in so many teams and the language problem is there and almost every team has encountered this. It happens that sometimes the requirements are coming in a way that we are sitting together trying to understand what they mean."

She also referred to the different speaking styles, speed and accents people have as a challenge for the development team members to understand the message

FRPK-IME-04-2013: "Some people have a difficult accent and some people talk a bit faster so it becomes hard to catch and understand what they are saying."

According to one of the developers the combination of lack of language skills and cultural awareness complicated the problems of understanding what was being said.

FRPK-MAI-03-2013: the language barrier is a hurdle because of the difference not only in the language but also the culture… you cannot understand many things their signals or responses in the way [they are meant]."

Lacking the required ability in English language, the offshore development had to rely on the proxy client who had excellent command over English and was well aware of the client's culture.

FRPK-OOI-04-2013: In almost every conference call [the proxy client] is involved …he knows how to deal with the client can speak English and can communicate with the client much better than us. The [proxy client] has been living there for more than 20 years and understands client culture, environment and work habits."

The offshore development team had to rely on the proxy client to clarify their requirements related queries and verify their understanding of requirements. One of the team leads reported that the offshore team members often noted down their queries during the client meetings to be later answered by the proxy client.

FRPK-ZAE-04-2013: "If we don't understand something we note down the questions and ask [the proxy client] later… right after every conference call or webinar we sit together, and [the proxy client] tells us 'this is what was explained and now we have to do this'."

5.8.3.2 Lack of Language Ability Affecting Negotiation Skills

The development team's lack of command over English language affected their ability to efficiently negotiate with the clients. To negotiate project scope and requirements changes with the client often the proxy client had to be brought in. One of the team leads shared an example of a past project where the client wanted the development team to entertain a change request and was not willing to pay for it. The development team knew that the change request required significant work and resources and it could not be 'accommodated' in the existing scope. They however did not know how to negotiate and deal with the client directly fearing that they might not be able to communicate properly.

FRPK-ZAE-04-2013: "we got the proxy client involved and he negotiated with the client and convinced them why all of those [changes] could not be accommodated without cost. He said 'this and this we can accommodate as a good will but for that we will have to charge you'".

5.8.3.3 Delay Due to Language Barrier in Client Interactions

Difference in language was a barrier for communication for the offshore development team members especially during client interactions. The company had introduced a hierarchy and protocol to be observed prior to engaging in client communications.

FRPK-DUI-04-2013: "Because you want a quality communication with the client…normally only the Project Manager talks to the client and in his absence a team lead would."

One of the team leads talked about the delay in putting their message across and understanding requirements caused by the hierarchy and protocol that the team had to observe during client communication. Not being able to understand the requirements from the client and discuss them directly meant that the team required additional clarification meetings with the proxy client and other team members that resulted in delay.

FRPK-IME-04-2013: "when we want to communicate during a meeting we tend to write it [our question or comment] on a paper or notepad and show it to the development manager and he conveys it further. It does bring in delay but we have to manage it."

The development manager however did not consider language to be big problem. He suggested all team members could communicate and the quality of conversation was possible because other key members were present during client interactions.

FRPK-DUI-04-2013: "When we have someone new who has to communicate and coordinate with the client and does not have so good English language skills in that case the problem is only the quality of communication but communication does happen because some of the key people are present, who can cover any gap in communication left by say an inexperienced resource. So I do not see that as a big problem."

The proxy client similarly considered that language difference was not a problem at all. He stated that the clients realize that English is not the first language of the development team and therefore are considerate while talking. He further added that since the team members were given accent training so that they can understand and speak with the client properly, so language difference was not a big issue.

5.8.4 Challenges due to Culture

Culture was not reported as a major challenge in the WIPS project by the participants however, there were a few issues that related to culture. The difference in culture occasionally hindered requirements understanding as well as posed some challenges for female team members working in the offshore development team. Similarly, local culture of the offshore team and their work ethics were sometimes not aligned with the client culture that caused some concerns for the development team.

5.8.4.1 Requirements Understanding and Culture

Based on past project experience one of the development managers talked about how culture could impact requirements understanding. He suggested that since usury (i.e. taking an exorbitant amount of interest in excess of the legal rate) is not considered permissible in their local culture people tend to avoid even businesses transactions that involve bank interest payment. This created an interesting situation for the development team since they were developing an online shopping cart application and integrating credit card payments (involving bank interest). Most of the team members did not have any experience or knowledge about how such application worked. Online shopping is an emerging trend which is not well understood or experienced in the Pakistani culture

(Nazir, Tayyab, Sajid, Rashid, & Javed, 2012). Therefore, understanding application requirements became an issue for that project.

FRPK-DUI-04-2013: "Culture itself is doesn't have a very big role in GSD …but if you develop an application that is more suited to the client's culture, they will automatically know the requirements of that application…but we need to get that information from them, as we cannot experience it [in our culture] … for example, initially when we were developing credit card applications, our team members did not know about online shopping, although they were developing shopping carts for the clients. They used to visit different shopping carts sites in order to learn from them but they lacked any personal experience of online transactions… we do not have the culture of shopping online and using credit cards so in terms of culture we sort of lacked that kind of understanding."

5.8.4.2 Local and Organizational Culture in Client Dealing

Cultural differences were highlighted as a challenge when dealing with foreign clients and having to modify some of the local cultural habits by the team members. One development manager considered that culture understanding is something that has to be learnt by their team; not only to understand the conversation and the context but also to have better relationship with clients.

FRPK-OOI-04-2013: "it's a matter of learning …. we have to learn how to deal with people in the West… how to greet them at the start of the meetings, or how to react when they make funny comment in the meetings these are quite different and vary from culture to culture."

One of the developers shared the cultural aspect of working with foreign client and having to modify local cultural traits to the client expectations and standards as below

FRPK-WAA-03-2013: "there is a difference between working with the locals and foreigners; they [the foreigners] are perfectionists. When we have given a time let's say that this work will be delivered on the 15th march then they expect to get it on the 15th, while with the locals if we have said 15th it may be delivered somewhere around the 25th, that's the difference we have to consider while working with them."

One of the developers complained about the problem in dealing with a foreign client as he considered that professional ethics and courtesy was not upheld by some clients in communications.

FRPK-MAI-03-2013: "most of us in Pakistan have this impression that they are far ahead in work or ethics but this is our thinking…they think even when they shout at me I will suppress myself thinking perhaps it was my mistake. I have noted that among us shouting is rare but the

client shouts normally in business to business communication. Professionalism and business ethics are not upheld and followed."

However, the development manager considered that there was no sense of inferiority or the tendency to be 'agreeable' to unreasonable demands from the client by the development team members.

FRPK-DUI-04-2013: "I do not see any submissiveness or arrogance form either side…we take client requests on merit, if it is within the time frame and within budget and cost we take it, otherwise we say no politely… as a culture we try to avoid plain blunt 'no' because we do not want to show any disrespect to the client."

5.8.4.3 Late Sittings and Working Women in Pakistani Culture

Late sittings and working overtime was an issue for the females working in the WIPS project in particular but in software companies in general as reported by one of the team leads. Late sittings in WIPS project often created problems for the females working in the organization. In the local culture, women were not allowed to stay till late in the offices.

FRPK-IME-04-2013 "Because there is a cultural difference … [in our culture] we don't like that females stay out or in offices late till 11:00 pm or 12:00 am. The neighbours mind that as well."

Late sittings and night shifts were not socially acceptable for females due to their safety as women and the negative impact of coming home late at night might have on their respect in society. Therefore, female staff had to leave earlier than their male team members. Although that was considered acceptable but it affected their team contribution and commitment. One of the team leads reported that due to the cultural constraints as well as security reasons females had to leave early.

FRPK-IME-04-2013: "we had a demo on last weekend, the three male members went home at 6:00 am in the morning and they worked all night long but I was a female in them I had to leave at 8:30 pm because I couldn't stay [in the office] later than that."

5.8.4.4 Culture and Team Participation by Women

One of the female members in the offshore development team discussed the issue of shyness for women in team participation. According to her, during meetings (both internal and external) female team members were often shy to ask question and share their views or make their point.

FRPK-IME-04-2013: "when it comes to our social norms… girls are shyer and keep themselves a bit behind… I personally feel that quite often when there is a team meeting and guys are discussing something then it is very hard to stop them… so the females have to raise their hands or make a gesture … otherwise they don't listen."

In a response to the question regarding females being submissive to their male counterparts she replied

FRPK-IME-04-2013: "if sometimes the facts and figures are not with you then you have to be submissive."

For the proxy client the cultural differences between the client and the vendor team was an added advantage. He stated that since their clients belong to a country which allows cultural diversity [the USA], so it was not a problem for them. Moreover, according to him, the offshore development site had the processes in place to reduce the effects of culture and they were also going through cultural trainings to bridge cultural gap. According to him the company placed extra emphasis on hiring good people from diverse cultural background and social circles to address cultural issues both with the client as well as within the organization.

FRPK-IAE-04-2013: "Knowing the culture is a plus. The client country is a diverse country that values and accommodates other cultures. Cultural issues are minimized through training and the processes in place. At the time of recruitment, we tend to find and hire good people from all social circles, with the right attitude [and t hose] who need work."

Table 5.6 summarizes the issues related to the culture difference between the client and the vendor team.

GSD Challenges--	**Challenge Description**
Challenges due to Geographical Distance	Distance and Dependency on the Proxy Client for Decision Making
Time Zone Difference	Time Zone Difference Late Sittings due to Lack of Overlapping Hours efficiency Time Difference as the Cause of Inefficiency and Stress
Language Difference	Problem in Understanding Requirements Lack of Language Ability Affecting Delay Due to Language Barrier in Client Interactions
Challenges due to Culture	Requirements Understanding and Culture Culture in Client Dealing Late Sittings and Working Women in Pakistani Culture

	Culture and Team Participation by Women

Table 5. 6 GSD Challenges Faced During RCM in the WIPS Project

5.9 Role of Collaborative Technologies Used

This section deals with the second research question of this study that relates to the role of collaborative technology in managing requirements change activities. To answer this question, the interviewees were first asked to provide their perceptions on the overall and general role of collaborative technology (CT) in requirements change related activities. Secondly they were asked about the role of collaborative technology in dealing with any critical incident they had faced during the course of the project. The participants were asked about CT and its role (both helpful and hindering) based on the six core principles of Activity Based Computing framework. Based on the answers to these questions the experiences and perceptions of CT's role in this project was analyzed using the thematic content analysis technique as discussed in Chapter 3.

In the coming subsections both the helpful and hindering role of collaborative technology are discussed respectively based on the perceptions of the research participants. Collaborative technology (CT) was blended into all the activities of requirements and change management in the WIPS project see Figure 5. 8. Requirements and change related activities were carried out using several collaborative technologies however each one of these technologies was considered important at its level and in its place. In fact, the role of collaborative technology was considered helpful by all the participants; something without which they thought collaboration in a GSD project wasn't possible. It was therefore hard for some of the participants to identify the single most important collaborative technology.

Between Pakistan and USA											
Collaborative Technologies and Tools for Requirements Change related Collaboration											
Name of Tool or Technology	Used In Activities										
	Change Reporting/ Feedback	Change Elicitation	Change Analysis & Review	Change Negotiation	Change Specification	Change Validation	Change Implementation	Change Testing	Traceability	Configuration Management	Review & Storage
Email	√	√	√	√ Exchanging and Negotiating over Specification Spreadsheets	√ Exchanging Specification Spreadsheets	√ Discussions & Feedback	√ Demo Review Feedback	√ Demo Review Feedback (internal & External Clients)			
Phone, Mobile	√	√									
Conference Calls (Lync, Live meetings), Webinars, Screen Sharing	√	√	√	√		√	√				
Electornic Artefacts (MS Word, Spreadsheets, Images, Diagrams etc.)	√	√	√	√	√	√		√	√	√	
Bug Reporting Tool or Ticketing System	√	√	√	√	√	√	√	√	√		
Shared Directory & Location					√ Temporary Storage & Review						
TFS										√ Code	
Share Point									√		
VSS									√		
Drop Box									√		

Figure 5. 8 Collaboration Technologies Used for Requirement and Change Related Collaboration between Pakistan and USA Sites

The collaborative technologies used in WIPS project was reported to have played a vital and mainly beneficial role in problem resolution. The participants perceived the available technologies to be 'sufficient' in providing support to carry out requirements and change management activities. It was noted during the interviews that almost all of the participants agreed that the existing collaborative technologies provided support for all the six core principles specified in the activity based computing (ABC) paradigm (Bardram, 2009; Tell & Babar, 2011, 2012). The ABC principles included collecting activity information, altering between tasks, suspending and resuming activity by individual users and at various connected systems, sharing activities among participating users and allowing an activity to resume itself by adapting and adjusting according to the context.

The participants also discussed some of the challenge with the collaborative technologies. The reported challenges were; having to upgrade to newer CTs because the expiry of product support, context switching between multiple CTs, lack of knowledge or expertise by the team members when using CTs and having steep learning curve for adopting new technology. Instead of a single collaborative platform that could provide all collaboration related support during the software development most of the participants preferred to have dedicated tools with specific task support over integrated tool support.

5.9.1 Helpful Role of Email as a Collaborative Technology

Working in a GSD environment, email was considered the most important collaborative technology by four out of seven participants interviewed for the WIPS project. It was considered by the participants as the lifeblood of communication without which collaboration in general could not even take place in a distributed environment. In reply to a question regarding the most important collaborative technology to manage requirements change in GSD, one of the software engineers replied:

FRPK-WAA-03-2013: "I think the most important collaborative technology is email, without email there cannot be any collaboration… even the CRF comes through email."

Another software engineer also had the same opinion and added that emails were especially important because they acted as a bridge between the development team and the client while collaborating over change implementation. He talked about the supportive role and importance of emails as collaborative technologies in combination with bug tracking tool called QCQA centre. He suggested that without this combination issue management would not have been possible.

FRPK-MAI-03-2013: If you do not have emails you cannot work at all, even the QCQA (software) is dependent on emails, no one goes and checks QCQA until an email is generated it and everyone then follows that thread ….it is a collaboration between multiple stakeholders… without emails nothing would be possible."

Emails were also considered very supportive for collaboration during change elicitation, analysis and validation. According to one of the engineering, emails were instrumental in communicating change related information and enabling shared understanding (between the development team and the client) regarding change requests. Information sharing through email [and attachments] enabled verification and validation of a change when shared between the client/proxy client and development team.

FRPK-WAA-03-2013: As the change request is coming in points, let's say [the proxy client] has asked for ten things in points one to ten we give feedback against each point by changing font colour. After that [the proxy client] responds and gives his feedback."

Similarly, one of the project managers also favoured emails to be the most important collaborative technology in a GSD environment.

FRPK-OOI-04-2013: "I think it is the email which most important… I think because we are here in Pakistan and our client in US so email is very very important in this scenario… email is very efficient and it is permanently records the message."

One of the team leads also considered emails to be a vital link between team members and with the geographically distributed clients. It was a preferred technology over phone calls because of its permanency and as a documented proof of communication.

FRPK-ZAE-04-2013: "We cannot work without email as it is the backbone or the life of collaboration within team and across sites…. Email is better because on the phone call there is no document with you, there is no proof that you what you have said, no record."

5.9.2 Hindering Role of Emails as Collaborative Technologies

There were no challenges shared by the participant related to the use of email as a collaborative technology. One of the participants, however, shared the limitation of emails suggesting that sometimes it was difficult to resolve an issue or to reach an agreement using email.

FRPK-WAA-03-2013: "when [the proxy client] says we need this [change] we respond through an email … if we are stuck in the same thread even after two or three email exchanges we arrange a conference call. We take the matter up and discuss it on the phone to clarify or understand it."

While discussing limitations of emails, another software developer suggested that when a single issue is discussed both in emails and phone calls it is hard to establish a link between them. The developers take notes while discussing an issue over the phone call, which they sometime have to link with an ongoing discussion in emails. They struggle to connect and align their understanding (often captured on a notebook or a note pad) with the email correspondence as well as phone call conversation and keep an updated record for both.

FRPK-MAI-03-2013: "Linking the conversation between telephone call and email is an area of improvement, where we have had discussion on email and phone calls we need some sort of bridging and collaboration, for example the conversation on the phone we take notes, in order to align our common understanding we need to link it with the existing discussion in emails."

Although email as a collaborative technology was reported to have very little problems however one of the development manager reported a limitation in the available features in one of the email software he was using.

FRPK-DUI-03-2013: "It would have been much better if the email client we were using could have provided me with another view of my inbox while I was in a search view so instead of switching views I could have opened a brand new inbox view so that I could go back to my search progress go back into the same context immediately. Once that view was closed to view another email, we had to go into the inbox and apply the same filters of search again and then

we had to go at the exact search date where we had left it so I think that is something that could be improved as part of the email client we are using."

5.9.3 Helpful Role of Team Foundation Server (TFS) as a Collaboration Technology

The offshore development site in Pakistan had recently started using Visual Studio Team Foundation Server (TFS) as a collaboration platform to effectively manage their software development lifecycle. TFS as a newly adopted team collaboration tool (moving on from *Visual Source Safe (VSS)*) was being used only for code collaboration and versioning functionality. Team members for the WIPS project were new to this technology and needed training to fully utilize the available features of the product.

FRPK-DUI-04-2013: "We have shifted to TFS…we are not using all the features of TFS because it was recently introduced in our organization so we have shifted only the code versioning part to it …we are working towards using it as a fully integrated project management solution for us in the future."

The benefits reported by the development team members regarding TFS are presented here.

5.9.3.1 Better Code Management and Collaboration Support

For the WIPS project the development team members shifted to TFS which according to them had better support for code management. The role of TFS as a code management tool was considered crucial; as per one of the developers.

FRPK-MAI-03-2013: "as far as writing and managing code is concerned, you cannot work without it [TFS], for example there would be no backups for the developed code; if TFS is not there then there will be some major problems."

The development organization was previously using Visual Source Safe (*VSS*) for code management but was facing some problems. One of the developers stated that the data backups were slow and they were facing issues with certain features in *VSS* such as lack of team collaboration support, multiple file check-in.

FRPK-WAA-03-2013: "We use TFS for code synchronization and code versioning… before that VSS was used and although we could do back up with VSS but that was quite slow, there were many other problems…it was slow and had problems with lock ins. In this TFS we have multiple check-ins and check outs if there is a file on which many people are working."

5.9.3.2 Issue Tracking and Management Support

The offshore development members found TFS to be more supportive of various requirements related activities e.g. issue tracking and management. The team members reported that Microsoft Visual Source Safe could not be integrated with the in-house developed bug tracking and management tool (QCQA) [20] therefore they found it difficult to log in and manage their bugs. Talking more specifically about some of the functionalities that were either not available in *VSS* or considered better in TFS one of the managers stated that;

FRPK-OOI-04-2013: "visual source is only for versioning not for other purposes. QA team could not log in bugs in VSS but in TFS they can …VSS did not allow to share screens…we can do a lot of things in TFS."

5.9.3.3 Requirements Management and Project Management Support

The role of TFS was considered important in collaborative development both for carrying out requirements and change management work. The development manager considered that there was no need for any dedicated requirements management tool support as TFS could provide support for requirements tracing, management and configuration management activities.

FRPK-DUI-04-2013: "TFS does all of this [requirements management, requirements management, requirements tracing] and we are moving towards using TFS for most of these tasks, but gradually …of course for requirements communication and collaboration that is being done through exchanging some design document and requirements documents and all those documents are already kept in our VSS as well as TFS which we can always trace back."

One of the team leads talked about the reasons TFS is being use as their new collaborative technology. She reported that it's the new and additional features such as task assignment and tracking as well as requirements and testing related activities that can be performed using TFS.

FRPK-IME-04-2013: "TFS is being used because it has additional support for project management…you have the option to assign tasks and keep its track…. requirements and testing related options are also there."

[20] https://msdn.microsoft.com/en-us/library/3h0544kx%28v=vs.80%29.aspx

5.9.3.4 TFS as an Important Part of Collaborative Technology Repertoire

The team members involved in requirements modelling and design found *TFS* to be quite useful. *TFS* could be easily integrated with the development environment such as *Microsoft Visual Studio* as well as *Microsoft Visio* [a modelling software] to allow collaboration for cross functional team members. Its integration with *Visual Source Safe (VSS)* to enable easy storage and retrieval for requirements and other project related artefacts was also considered very useful by the participants.

FRPK-WAA-03-2013: "TFS is tightly integrated with visual studio and you can link it through browser from TFS… is tightly integrated with MS Visio…integration with VSS can also be done, let's say if you want to see a file in VSS by browsing, so you would need a client for VSS. So you can connect visual studio to TFS."

Instead of singling out the most important collaborative technology, the developer considered that CTs act as components that combine to produce the output or achieve the goal of collaboration. Sharing the same views one of the development managers stated about the role of TFS in the repertoire of collaborative technologies.

FRPK-DUI-04-2013: "each and every [collaborative] technology is required at its own level I mean you need TFS for code collaboration and management, you cannot perform development work without it …for communication you need Live Meeting and Lync… I mean everything has got its own importance so it is hard to pin point a single one."

5.9.3.5 TFS as a Technology with Potential to Improve Productivity

The offshore development team members saw TFS as a collaborative technology that could improve their collaboration and performance. The development manager considered that they could move on to using all the offered collaboration support in future to benefit their organization.

FRPK-DUI-04-2013: "we do feel that we need to use other features provided by TFS and Microsoft project server as well to better manage the project and code release process, we are going to that …we have so many areas we can improve by using these technologies."

Similarly, one of the team leads suggested that if TFS was to get fully implemented both for project management and requirements related activities the overall productivity and efficiency could be improved.

FRPK-IME-04-2013: "If you are trying to manage the project properly and your software facilitates that by providing you ease in tracking different things, obviously that has a positive impact on your performance."

5.9.4 Hindering Role of TFS as a Team Collaboration Technology

The development team members did mention some challenges related to using TFS for code collaboration and as a collaborative technology.

5.9.4.1 Learning Curve and User Training Requirements

The development team faced problem in using TFS due to its complexity and the team's lack of experience and training as reported by one of the development managers. He considered that training the staff for using the newly adopted collaborative technology (TFS) was the only option available to solve the problems faced.

FRPK-DUI-04-2013: "because our new development team members don't have experience with TFS at times they make blunders so we have to train them… this is something that can happen with any technology if you don't know how to use the technology the only remedy is to train people."

Similarly, one of the team leads talked about code management issues when inexperienced people were using TFS with its multiple checkouts functionality.

FRPK-ZAE-04-2013: "Due to staff inexperience we had to disable multiple check outs in the project and if we had not done it we would have been in a mess."

The team lead considered that TFS was problematic only when the people using it did not have enough knowledge or training on how to use it.

FRPK-ZAE-04-2013: "I have not found any problem with TFS, until and unless you do not know how to use it."

One of the development managers reported that even with training the desired results of effective use of the technology were not always possible.

FRPK-DUI-04-2013: "we did train our staff but we don't take any formal examinations [after training]… even after the training some people do not follow the guidelines…it is the same as training someone to do spell checking before sending out an email and they still send it without checking their spellings."

5.9.4.2 Having to Move on to TFS

One of the project managers shared their thoughts about collaborative technologies and the reasons for adopting new collaboration techniques. He thought that upgrading to a newer version of collaborative technology was not always because of its problems. A newer technology might provide added features but the decisions were made either in light of organizations technology partnerships (e.g. with Microsoft and Oracle) or at the discretion of higher management. The team members did not have any say in that matter and had to learn any technology that was adopted by the company. Providing reasons for adopting TFS and leaving *VSS* as their collaboration technology he said:

FRPK-DUI-04-2013: "the reasons for leaving VSS were number 1, this [TFS] is a better ending technology, Number 2 Microsoft itself dropped support for VSS. It was becoming deprecated product."

FRPK-OOI-04-2013: "[we shifted to TFS] because Microsoft has stopped releasing new versions of VSS from 2005[smiles] so we had to shift."

5.9.5 Helpful Role of Microsoft Lync as a Collaboration Technology

GSD Inc. used Microsoft Lync for both within and cross-site business communication and collaboration. The participants considered Lync as a CT that provided significant support for work collaboration especially around requirements. Lync as a unifying communication and collaborative technology offered several capabilities such as instant messaging, audio & video calling, conferencing and content & screen sharing. The participants reported the content sharing features such as file transfer and video conferencing were not used very frequently.

5.9.5.1 Screen Sharing for Requirements and Change Understanding

Screen sharing feature in Lync was used for local as well as global collaboration. This facility was particularly useful in requirements and change understanding and for validation once the changes were implemented. While working with remote clients often the development team had to show and pin point the problem on the screen to reach a common understanding of an issue or the bug reported.

FRPK-WAA-03-2013: "we share the screen and show him where the problem exactly is… in OCS [collaboration technology used previously] screen sharing wasn't there and we had to use software called Team Meeting for collaboration work. Now we use MS Lync across sites."

One of the developers stated that this feature also allowed developer the freedom to work from their own station rather than having to move around and go at a different station to visually verify problems or resolve them.

FRPK-ZAE-04-2013: "I do not even have to leave my seat and I can simply do a screen share [with others]."

Working in a global software development environment meant having to deal with the lack of face to face activity and the next best thing for the participants was to have some communication and collaboration technology which could allow them to communicate and work together.

FRPK-ZAE-04-2013: "Lync facilitates us to do meeting when someone is not present face to face."

One of the developers reported that Lync was only used between the development team at Pakistan and the proxy client in the USA. To communicate with the client other means were used such as emails, phones and webinars. He considered without the frequent interaction that takes place with the proxy client through Lync or the telephone the communication between the two sites would virtually end.

FRPK-MAI-03-2013: "We use Microsoft Lync but that is used between [the proxy client] and [development site at Pakistan] …for interacting with the client we have phones and emails and webinars, where we can share the screen…if you take Lync or telephone out we will not be able to communicate with [the proxy client]."

5.9.5.2 Lync as a Unifying Communication and Collaboration Technology

The chat facility provided by Lync was also used by one of the development leads as a permanent storage place for communication when assigning tasks to other developers. He told that chat history feature allowed him to refer to previous communications between him and the developers in case there was a dispute in task assignment.

FRPK-ZAE-04-2013: "So in some places we can even get certain things from the chat as well. As I work and talking with a junior and if there was something which got discussed and she/he is not accepting it that we discussed it, then we can refer back to the chat, which I can check from the chat history as well, as it is getting logged".

The organizational user accounts were also set up in a way to facilitate integration with Lync making it easier for the team members who did not have to do extra logins to use Lync for their work collaboration and communicate

FRPK-WAA-03-2013: "Microsoft's Lync which is a messenger that is integrated with our organization's official account…, it has all the features like a messenger, you can also share screen with it. Let's say you are making the client understand some problem, sometimes he asks us to share the screen."

5.9.6 Hindering Role of Microsoft Lync as a Collaboration Technology

Although Lync as a communication and collaboration technology was mainly reported to be beneficial by the participants some challenges were also identified while using Lync. The main challenges were: adopting Lync a new CT (requiring a steep learning curve), misuse of certain features (such as chatting) and the lack of client access to Lync.

5.9.6.1 Having to Use Lync as a Collaboration Technology

Being a technology partner with Microsoft it was the organizational policy to move ahead with the new technologies as they get developed by Microsoft. A decision in which the development team did not have any say. He added that there are so many technologies being adopted that it has become a routine rather than a problem.

FRPK-WAA-03-2013: "We use Lync as our messenger … earlier it was OCS…there were no problems with OCS but because Microsoft has made this new thing so now we are using this one…its [the proxy client's] choice in which we have no say."

5.9.6.2 Difficulties in Arranging Conference Calls with the Clients

The development team had to face the problem of having to use a Live Meeting session rather than using Lync for collaborating with the client, since the client was not added to their Lync contact list which was a problem. The development team used a conference call account for Microsoft Live Meeting which they shared with the client at the time of conference.

FRPK-DUI-04-2013: "But if we have to do a meeting with the client we host a Live meeting session on our Microsoft account because they are not added to our Lync contact list…we have a conference call number that we give it to our clients and they can call in and put in their pin number and through that they can login and take part in live meetings."

5.9.6.3 Misusing Lync's Chatting Facility during Office Hours

One of the development leads considered that sometimes the downside of having chat facilities available to the local team members was that they used it for non-work related activities and misused it.

FRPK-ZAE-04-2013: "So if you talk on Lync then some people of course do 'chatting' on that not for official purpose and they are misusing it. This has happened already."

5.9.7 Helpful Role of Bug Tracking Software (QCQA) as a Collaborative Technology

GSD Inc. had an in-house developed Quality Control and Quality Assurance (QCQA) centre as their issue tracking and management system. QCQA software was considered helpful in carrying out bug related communication and collaboration between the development team members, QA and the clients.

5.9.7.1 QCQA as a CT and a Bug Tracking Tool

Stakeholders across globally distributed sites could send their feedback, report and track progress of the identified bugs using QCQA. The client also used QCQA to report and send their feedback and new requirements whenever they were provided with a software release for acceptance testing.

Talking about the online interface for the QCQA software and collaboration role played by the technology between the client, development team and the proxy client one of the developer said

FRPK-MAI-03-2013: "QCQA center is a web based application and a kind of technology through which everyone; the developers, the management (QA and Development) the client and proxy client can collaborate, report bugs and maintain bug history … the client also sends bugs or requirements through QCQA found identified during acceptance testing."

QCQA was integrated with a project monitoring tool to provide the management a snapshot of work progress and to track both requirements and change development work across all projects.

FRPK-DUA-11-2013: "Now there is a monitor which has been developed for our QCQA centre that provides us an overview of the bugs and from which project are they coming from. We can find out this information through this single-view monitor to manage the projects."

5.9.8 Hindering Role of Bug Tracking Software (QCQA) as a Collaborative Technology

There were a few issues identified while using QCQA by the development team members as well as the client. The issues related to QCQA were lack of detail required for meaningful reporting for issues/bugs assigned to individual developers, project status reporting capabilities and the constraint to export bug reports only using Excel sheets.

5.9.8.1 Misuse of QCQA to Record Unverified Effort Hours

One of the developers complained about the lack of control in task assignment, monitoring in QCQA. He considered if a commercial software was used then developers could not exploit the flaw in QCQA and enter unverified effort hours to portray themselves as being more productive than others.

FRPK-WAA-03-2013: "We have developed QCQA which is a bug tracking software but also used for task management …if we had implemented a proper task management system we could collaborate and work through it better, knowing who is performing what task and how much [effort] has been put in. Here people keep putting [effort] hours without much evidence."

5.9.8.2 Lack of Support for Issue Monitoring

Another developer talked about the lack of client support for task monitoring and interaction in the QCQA center [paraphrased].

FRPK-MAI-03-2013: "as we are talking about client collaboration between multiple stakeholders, in my opinion QCQA is not good from the perspective of client interaction…our QCQA does update the users relevant to a new thread launched however it does not treat task and bugs separately…it does not allow users [the developers or the client] to monitor progress on a particular task … it does not say how much work is done, how many issues have been resolved in that. It does not provide status update or progress and the information is received from QCQA only when a task is marked as 'done' then you receive a notification…here we have a simple straight forward tech which can be improved."

5.9.8.3 Inadequate Status Reporting Functionality

The reporting functionality in QCQA was very basic reports which could only be exported in a very basic Excel format and without much detail. Therefore, the client had to be provided with bug reports with additional information using Excel sheets which could then be used for further collaboration during testing and bug fixing cycle.

FRPK-DUA-11-2013: "So when we have to generate a report in order to send it to the customer then we have to generate it in Excel for easy access and use."

5.9.9 Helpful Role of Spreadsheet as Collaborative Technologies

In the WIPS project, spreadsheets were used as collaborative technologies (CTs) to work on and store requirements and design related artefacts. These artefacts include requirements and design specifications, enhancements, requirement change logs, bug reports and annotated screenshots. Often these artefacts were embedded in the

spreadsheets and were shared and exchanged through emails for cross site communication and collaboration. The cross-site stakeholders found spreadsheets as a very useful collaborative technology to share and collaborate over requirements and changes as evidenced below

5.9.9.1 Support for RE and Collaboration Activities

Asynchronous collaborative activities for requirements engineering and change management were facilitated by spreadsheets. The collaborative activities supported by spreadsheets included analysis, design, specification, negotiation, validation bug reporting and bug resolution managed through commenting on the spreadsheets. Stakeholders engaged in these activities could visually contribute towards the final form of the shared requirement artefacts (e.g. design specifications) by using timestamps, site information and colours (to identify individuals or sites) while commenting and exchanging them through application sharing technologies such as MS Lync (Hussain & Clear, 2014).

5.9.9.2 Support for Traceability Activities

For the WIPS project spreadsheets were also used to support traceability. To support traceability design specification documents contained a column named '*Design Document Page #'* to link a particular specification with other requirements. This column had reference page numbers to the individual a) change items in design screens, b) requirements sections or c) to the associated use cases. This practice was adopted by the project team to them help with their traceability efforts (See Figure 5. 9). Although a traceability matrix (TM) existed it was not kept up-to-date due to documentation effort required for continuously changing requirements throughout the project.

5.9.9.3 Spreadsheet as a Supportive CT for Global RM Practices

The spreadsheet based artefacts had a 'content-evolving' nature and aided the team members for their requirements and change management related activities. The client and development team members used various font colours to identify and distinguish one particular role, space or location from another. The colours also aided quick identification and retrieval of the desired information. Timestamps were used to enable traceability among messages and quick retrieval of the most current information. The content of the spreadsheet artefacts underwent progressive sets of changes, revisions and evolutions

over time, but was maintained as the current live version. It also acted as 'discussion lists' consisting of comments and updates from the team members involved in the activities such as specifying, analysing, negotiating, verifying and validating requirements (See Figure 5.10).

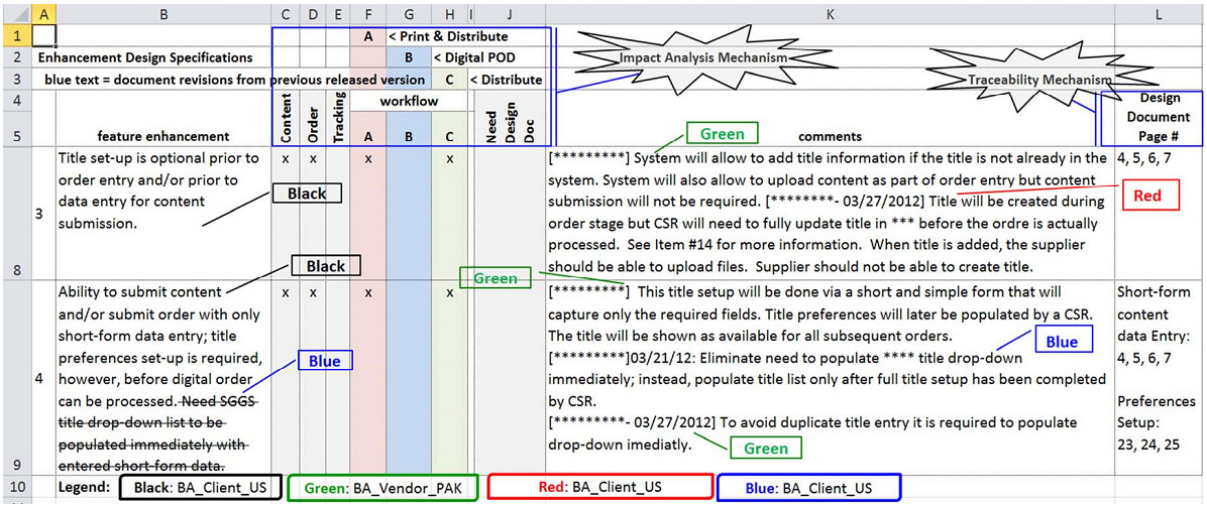

Figure 5. 9 Role of Spreadsheets as Collaboration Technologies for Traceability

5.9.9.4 Usefulness of Spreadsheets as CTs for Requirements Collaboration

The development manager was asked regarding the purpose and usefulness of spreadsheets as the main artefacts for requirements related artefacts and collaboration the design specification, his reply is summarized below

- *It is an easy to use tool for which you do not need to set up any infrastructure.*
- *The client might have software licencing issues if we use any other commercial tool for requirements related artefacts*
- *Spreadsheet software such as MS Excel are immediate tools that do not require additional training to use for requirements related collaboration (e.g. sharing, commenting)*
- *Readily understandable and easy to use for collaboration*
- *Easy for making modifications in the document such as column additions, commenting, using coloured test and also provide basic drawing facilities*
- *They also provide options for filtering, sorting and searching*
- *Tools need to be similar across sites or have outcomes which are compatible (exporting and importing outcomes is easy) which is afforded by spreadsheets.*

He argued that spreadsheets were not easily replaceable by any other commercially available tools as other tools may also provide the features discussed above but they may not be as easy to use or may not be on everyone's finger tips

5.9.10 Hindering Role of Spreadsheets as Collaborative Technologies

Use of spreadsheets as collaborative technologies has been reported as positive and helpful by the research participants. However, although not reported by the participants the analysis of artefacts revealed that a high number of design documents (at least 36) and prototype modifications (screenshots annotated by users) made it difficult to gain a consistent view of requirements as well as changes. Similarly, the effort to maintain and update various versions documents while not allowing Excel tools to proliferate isolated and contradictory data pools may have been a significant challenge. Table 5.7 summarizes both the helpful and hindering role of collaboration technologies.

Collaborative Technology	Helpful Role	Hindering Role
Email	• Reported as one of the most important collaboration technologies • One of the most relied and heavily used asynchronous collaborative technologies. • Extensively used as a reliable tool to share requirements and change related artefacts (specifications and design) and feedback on artefacts. • Containers and carriers (as attachments) of requirements and change related documentation and feedback/clarification)	• Not a good technology to resolve issues • Difficult to manage and link with communication such as phone calls, chat and audio conferences
TFS	• Provided support for better Code Management and Collaboration • Provided issue tracking and management support • Helped with project management and requirements management activities in an organized way. • Played its part in the repertoire of collaborative technologies. • Helped as an information retrieval facility available for cross site communication and usage • Offered potential for improvement in the organizations productivity and performance	• Demanded huge learning curve • Usage by people lacking appropriate training caused problems in code collaboration and management • A technology not adopted as a choice by the team members rather something that was forced on them • Did not allow single item (e.g. a file) sharing in multiple projects
Microsoft Lync	• Facilitated distributed collaboration in an affordable manner	• Preference of using less rich media for communication (only voice not video)

	• Facilitated various requirements related synchronous meetings for review, analysis, design, specification and validation activities. • Found most effective when used with the screen sharing feature that allowed sharing of requirements artefacts such a bug.	• A technology not adopted as a choice by the team members rather something that was forced on them • Arranging conference with client who were not using Lync. • Misuse of chatting feature for personal use during office hours
QCQA	• Provided issue/ bug tracking facility • Helped in testing and issue status reporting • Provided management support for project monitoring • Helped in project status reporting • Facilitated issue related collaboration • Provided issues related collaboration information for knowledge workers • Helped in reflection on the previously logged issues and for future planning	• Used for task management by the organization but had poor support • Unable to track performance or effort spend by individual developers • Not considered good for client interaction due to lack of inbuilt task communication and tracking facilities • Had very basic reporting functionality • Reports could only be exported in excel format and in very basic format without much detail.
Spreadsheets	• Supported all requirements related activities while serving a summarizing and communicative across distributed stakeholder groups • Complemented special purpose collaboration technologies (e.g. ticket tracking systems, IDEs etc.) to cater for the distributed collaboration needs • Enabled discussion and supported artefact editing when used in combination with as Lync with active screen sharing feature. • Allowed asynchronous use of spreadsheet use was supported through emailing files between sites • easy to use and immediate tool that provide options for filtering, sorting and searching	• Proliferation of isolated • and duplicated data pools (bug report, design docs, specifications)

Table 5. 7 Summary of Helpful and Hindering Role of Collaborative Technologies Used in the WIPS Project

5.10 Chapter Summary

This chapter has provided the overview and contextual analysis of the second GSD project (WIPS) studied for this research. It has also presented an analysis of the prescribed model for RM and has highlighted the issues found between the prescribed and practiced activities for requirements and change management. Furthermore, the chapter presented

the analysis of the model in practice using the lifecycle activities and discussed the identified challenges faced by the practitioners in each lifecycle activity. Finally, this chapter has analysed both the helpful and hindering role of collaboration technologies used for the GSD projects.

Based on cross-case comparison, the next chapter discusses the findings reported for each of the studied cases in this research.

Chapter 6. Discussion

In this chapter the findings from the two cases studied are compared and contrasted. They are then discussed with regard to the research questions posed in this study.

Section 6.1 discusses the eight newly identified challenges based on the findings from the two studied cases for this thesis. The impact of these challenges on various requirements management activities is discussed and the results are situated in the existing GSD and COTS literature. In Section 6.2 a model is provided that captures the newly identified GSD challenges as well as the previously known challenges that were experienced in the two studied cases. The effects of these challenges on RCM activities are briefly discussed and summarized. Next in Section 6.3 the helpful and hindering roles of collaborative technologies are discussed that are based on the participants' perceptions. Finally, in Section 6.4 a synthesized model of actual change management practices is discussed based on the findings from the studied cases along with reflections on existing RCM models found in GSD literature. Research evaluation is then provided in Section 6.5. The chapter ends with a concluding summary in Section 6.6.

6.1 Newly Identified Challenges impacting Change Management Activities in GSD

The analysis and findings from the two studied cases in this research revealed eight new challenges that impact RCM activities in GSD. These new challenges and many other confirmatory challenges experienced in the studied cases are discussed in the following sections and are summarized later in Section 6.2. The detail of the identified challenge is provided here on the basis of its impact on a particular case studied (e.g. Case 1 or Case 2) and the impact is then compared with the other studied case to provide a coverage of the challenge in both cases.

6.1.1 Immaturity of the Product

Both cases analysed in this research were centred on modifications to existing products. When compared to each other, however, these products were significantly different in terms of their degree of feature completeness, available supporting documentation (to gain product knowledge) and the extent of customizations or changes required to fit the customer organizations' needs. These factors; taken as a whole were used as a measure of the maturity of the product or solution. The relative maturity or immaturity of the product and the corresponding high and low levels of required customizations significantly impacted elicitation, analysis, negotiation and implementation activities of

a requirements change. The main challenges posed by product immaturity on managing changes in requirements are discussed below.

6.1.1.1 Elicitation Challenges due to Feature Incompleteness

The product in the CICP project (Case 1) was a newly developed research management COTS package which had incomplete functionality in some major areas. As a result, the product required major development work (as opposed to minor customisations). The challenges experienced in Case 1 (NZ and USA) in working with an immature COTS package with missing features related to: identification of existing and missing product features, mapping of user requirements with the existing features and defining customization requirements for COTS package. Product immaturity mainly affected the elicitation, analysis and design phases during package customization in the CICP project.

Immaturity of the COTS product became one of the major causes of misalignment between the available product features, organizational processes and user requirements. Feature incompleteness and lack of product knowledge (due to minimal documentation) made product understanding and identification of matches and mismatches between product offerings and organizational requirements challenging. These challenges were compounded due to the complexity of the COTS product and the lack of product knowledge possessed by the client BAs responsible for elicitation. These observations are consistent with the findings of Lauesen (2002), who argue that understanding a large and complex software package even through an exploratory review is difficult for client organizations who generally lack product knowledge. This leads to difficulties in feature identification and mapping through product reviews (as witnessed in Case 1).

Difficulties in the elicitation of customization requirements experienced in Case 1 are consistent with the observations made by Ruhe (2003), and Lauesen (2002). The studies confirm the findings of this research as both authors argue that in large and complex software packages, incompleteness of information leads to product uncertainty regarding the existing features of the packaged software for the clients. As a result, feature evaluation, identification (and mapping to existing organizational processes) becomes a challenge for the client organization.

Furthermore, mapping product features with organizational process during elicitation can also be more challenging when the vendor promised features either do not exist or are incomplete as experienced in Case 1.

These findings are consistent with the recommendations by Boehm (1999), who has warned buyers to be wary of advertised marketing claims and vendor promises regarding COTS capabilities which may or may not be real. These elicitation challenges in Case 1, had a ripple effect on downstream change related activities such as problems during scope negotiations, change implementation, product testing. These findings are consistent with those reported by Morisio et al. (2000), who posit that COTS customization projects are obliged to follow non-traditional project development approaches and have to place additional effort into requirements elicitation and testing. Resolution of these challenges required significant communication and collaboration overhead which adversely affected the project budget and schedule.

6.1.1.2 Elicitation Challenges due to Inadequate Product Documentation

In Case 1, lack of supportive product and user documentation was an impediment to requirements elicitation. Identification of missing features and verification of existing ones in the face of limited documentation was challenging for the customer organization. It impeded common understanding of requirements and introduced further challenges for RCM activities. These challenges are consistent with the findings reported by Tarawneh et al. (2011), who suggest that absence or inadequacy of documentation is one of the factors that results in faulty evaluations of packaged products.

The analysts responsible for elicitation struggled to gain consistent understanding of the product due to inadequate and missing product documentation. They had to spend significant extra effort to explore the packaged software to identify existing workflows and features. They also had to rely on the vendor expertise to explain or show product capabilities and functionalities which were easily identifiable by the BA in Case 2. Lack of documentation and available product knowledge did not allow the client to establish the requirements baseline. In the absence of a clear baseline testing the product became significantly challenging as the testers did not know what baseline of functionality were they testing the new release against. Similar findings have been reported by Benguria, Belén, Sellier, and Tay (2002), who suggest that since a large part of evaluation is carried out using vendor provided product documentation which is often insufficient and always requires extra effort during COTS evaluation by the users. Morisio et al. (2002), note that project leaders in a COTS based development project have to deal with the challenge of unavailable, incomplete or unreliable documentation on the COTS product. They report

that testing a COTS product is also challenging as it has to deal with large modules where virtually no source code or documentation is available.

In contrast, adequate product documentation was seen as a contributing factor supporting common understanding of requirements and relatively stable requirements in Case 2. Sufficient product documentation was made available to facilitate product understanding in Case 2 and the practitioners involved did not report any of those challenges mentioned in Case1. These findings confirm COTS implementation reports found in the literature which suggests that the access to adequate product documentation (across user, system and application level documentation) is very helpful when acquiring, evaluating, customizing packaged software (Baker, 2002; Gross & Ginzberg, 1984; Baharom, Yahaya, & Ahmad, 2011).

Morisio, Seaman, Parra, Basili, Kraft, and Condon (2000) report COTS based development projects must accept vendor reliance and lack of quality product documentation (being unavailable, incomplete or unreliable) as a common condition and a challenge. The findings from Case 1 highlight several challenges due to the virtually non-existent documentation, in stark contrast to Case 2 where adequate product documentation aided the RE, implementation and testing activities. These observations reinforce long-held findings regarding the general importance of product documentation as reported by Baker (2002) and Gross and Ginzberg (1984).

6.1.1.3 Negotiation Challenges due to Intensive Customizations

In Case 1, the majority of the feature misalignments were identified after the selection of the product and during the customization phase. These mismatches between the features offered and the organizational needs (called mismatch) led to major modifications in the contracted requirements and therefore the product itself in Case1. Due to the high cost of adaptation these desired customizations had to be negotiated with the remote vendor. These proposed changes had to be carefully reviewed and evaluated both by the client and the vendor to avoid challenges after implementation. However, these reviews necessitated continuous remote negotiation, prioritization of requirements and scope adjustments creating challenges such as unanticipated communication coordination overhead, inadequate knowledge management and transfer, delays, and bad client-vendor relationship and other downstream change implementation and testing activities.

Challenges related to negotiation were exacerbated by the globally distributed nature of the project. Negotiations with the remote vendor regarding the identified customizations

in Case 1 were difficult due to collaboration barriers introduced by distance, cultural diversity, time differences and inadequate communication. These additional challenges to collaboration in a GSD environment as experienced in Case 1 are consistent with the findings by Damian (2007), as well as Noll, Beecham & Richardson (2010), who report that distance creates barriers for collaboration especially when collaboration is related to requirements. In Case 2 (Pak and USA), the existing product was complete and mature with a stable architecture, and therefore, feature mapping and identification of customizations was much easier as compared to Case 1.

6.1.1.4 Implementation Challenges due to Feature Incompleteness

In Case 1 (NZ and USA), emerging user needs and the product's immaturity meant that the numerous customizations were required. Many of the identified changes in the selected COTS product had to be carried out at the application's architecture level that resulted in breakage of product architecture. The COTS-based product in Case 1 was promoted to be flexible due to its resource oriented architecture of loosely coupled objects by the vendor therefore the product therefore could easily accommodate configuratory changes and customizations. Continuous requests of significant software intensive changes in the product however, had a negative impact on the product architecture as well as on the cost and schedule of the project. This observation aligns with the views of (Ruhe, 2003) who suggested that often packaged software customizations that are initially believed to be configuratory eventually become major changes affecting the product architecture and contributing significantly to customization costs.

The project in Case 1, kept on experiencing high frequency of software intensive change requests (due to product incompleteness and customer demands) which were adding time and cost to schedule and project budget respectively. The high cost of customization then made the client organization forego many important features they desired. The client on many occasions had to decide to modify their existing work practices or live with the shortfalls in the existing product not only to save time and cost but also to avoid implementation and testing related challenges. These findings are consistent with the general wisdom in COTS customization projects where modifications to the packaged software are generally discouraged as they are costly (in terms of development effort) and increase implementation risks (Sia & Soh, 2007).

The findings regarding feature compromise by the client also resonate with those reported by Hong and Kim (2002). The authors suggest that customer organizations faced with

‘feature-process’ misalignment weigh costs and benefits of customization and often choose to adapt their processes to the COTS package functionality, live with shortfalls of the purchased package or institute workarounds instead of heavily customizing the software package.

However, because of the incompleteness of the COTS package the client had to deal with development of intensive customisations which often required architectural changes. These misalignments represented the product’s incompatibility with the fundamental structure of the client’s business process reality. Product customization due to missing and incomplete business rules and key entities was required to make it compatible with the business requirements. However, these customizations had widespread implications (increased development cost, delays and quality concerns) for the studied project in Case 1. Sia and Soh (2007), have reported similar findings and suggested that ‘deep-structure misalignments’ are discouraged due to costly development efforts and the increased implementation risk by introducing bugs, adding complexity, making integration more difficult.

Product architecture related challenges that surface due to the working in a GSD context are reported in the literature (Catalado & Herbsleb, 2011; Herbsleb, Paulish, & Bass, 2005; Herbsleb & Mockus, 2003; Noll, Beecham, & Richardson, 2010). However, the architectural challenges witnessed in Case 1 due to product immaturity and its affect on product architecture resulting software intensive changes implementation are different from those generally reported in GSD literature. While prior studies have emphasised the challenges due to poor task allocation and coordination among distributed development teams as architecture related challenges, findings from Case 1 highlight the challenges of additional complexity, integration, cost development and testing resulting from an immature architecture.

The findings of Sheu et al. (2004), suggest that COTS package implementation can be challenging in a GSD environment because of differences in language, culture, politics, government regulations, management style, and labour skills. The authors suggest that these factors impact software implementation practices. Findings from the two cases studied in this research however, brings to light particular challenges (and their impact on various RCM activities) of customizing an incomplete COTS product for a specific client in GSD projects.

6.1.2 Lack of Product Knowledge

In one of the studied projects (Case 1), the client BAs tasked with requirements engineering activities were neither domain experts nor did they have enough product knowledge to conduct an effective product review. As contracted resources with limited domain knowledge they were also hindered in their ability to accurately understand and model some of the client's existing business process workflows and procedures. In addition to that the level of understanding required to review the COTS package and understand the functionality enough to appreciate the implications of adoption was also minimal in the client BAs.

Understanding and trying out large COTS applications in an exploratory fashion are often inherently complex (Lausen, 2005) however, lack of product knowledge and domain expertise of the client BAs made this work even more challenging. These identified challenges are confirmatory in nature as they have been widely reported in the packaged software evaluation, customization and implementation literature – for example (Gattiker & Goodhue, 2000; Hong & Kim, 2002; Soh, Kien, & Tay-Yap, 2000).

The remote vendor had limited capacity to understand and explore the clients' business processes sufficiently due to distance and the resulting lack of interaction with the client stakeholders. The vendor therefore could not transfer adequate product knowledge in a timely manner by highlighting critical areas possible feature misalignments and necessary customizations to aid the client's product understanding. The challenges in COTS product customizations align with the findings reported by Dittrich et al. (2009), who report that having adequate knowledge and understanding about the packaged software is the main challenge in customization. Highlighting the importance of knowledge related challenges Dittrich et al., (2009), further posit that even very experienced developers must consult experts on specific package modules to gain product understanding.

Inadequate product knowledge and its negative impact on the requirements elicitation issues identified in this case study is in addition to the challenges posed by geographical, temporal, and socio cultural distance and inadequate communication as reported by Damian (2007) and Nurdiani et al. (2011).

In Case 2, lack of product knowledge did not pose significant problems as some of the project team members had worked with the same client on a previous project dealing with the same product. Furthermore, gaining necessary knowledge about the product was also easy because 1) the existing product (and its related information) was online and publically available 2) comprehensive documentation and data was available for the

existing product, 3) the onshore proxy client was a domain expert and the development team had access to the expert consultants that were hired by the client to support the offshore development team.

6.1.3 Powerful Proxy Client

In Case 2, the CEO of the vendor organization acted as BA and proxy client. The role is comparable to boundary spanners, brokers, intermediaries or liaison who represent the vendor organization in client communications, facilitating awareness and knowledge management in distributed collaboration as reported by Manteli et al. (2014). The proxy client in this case was a 'tech-savvy' domain expert with over a decade of working experience in the vendor organization. He also had excellent communication skills and a good understanding of the socio-cultural and business norms of the client country. The aforementioned qualities, influential position in the vendor organization (being the CEO), and a successful tenure in this role earned him the title of an 'expert'. Such a position also provided him with additional powers as compared to other employees in the vendor organization.

While the above mentioned qualities enabled him to effectively negotiate the complexities of the project and its evolving requirements, it also introduced a "power asymmetry" mirroring that of the project itself in the global East/West setting (Ravishankar, Pan, & Myers, 2013). The misuse of the proxy client (vendor BA's) powerful position in the vendor organization (Case 2) often resulted in many challenges for the development team. The additional challenges were, dealing with excessive informal change requests, difficulties in negotiation of effort estimates, issues in requirements and change verification and misrepresentation of client requirements (Hussain & Clear, 2014).

6.1.3.1 Power Asymmetry a Challenge in Negotiation

The difference in power resulted in challenges for the development team (Case 2) especially while negotiating estimated time and effort with the proxy client for implementing requirements or changes. It was hard for the development team members to argue with the powerful role of the company owner / CEO. Often the offshore team members' proposed estimates for change implementation effort and time were cut back after discussions by the proxy client who had better negotiation skills. This caused the team members to spend additional effort to implement changes in requirements at their own expense through late sittings which almost always were without any compensation.

The developers were often given the justification that such reductions were part of the 'business strategy' in order retain or appease the clients. The development team members however felt exploited and they occasionally expressed frustration regarding the negotiation outcomes with the proxy client.

Case 2 highlights the challenges associated with the greater autonomy and authority of the intermediary role (proxy client) that resulted from his influential position in their organization, his expertise and network links. The unification of these properties in the proxy client's role shaped his autonomy and authority and allowed him greater role empowerment which created challenges in distributed collaboration. These findings are in contrast with those reported by (Perrone, Zaheer, & McEvily, 2003) who conclude that providing greater autonomy to a boundary spanner's role enhances the trust of the other party in client-vendor relationships. The findings regarding onshore boundary spanner or proxy client's power asymmetry and the related challenges reported in this research have not been extensively explored in GSD literature.

In Case 2 the interviewees (mainly at the managerial level) mentioned having an occasional argument with the proxy client and showed some frustration about the process of effort estimate negotiation. There was however, no mention of any resistance against late sittings, and uncompensated effort or power asymmetry. Lack of resistance regarding the challenges resulting from power asymmetry cannot however be considered positive or acceptable. It is argued that resistance is not an essential indicator that power was not exercised (Sauer, 1993, p. 48). Power can be exercised in subtle ways to even prevent an issue reaching an arena in which it can be contested (Bachrach & Baratz, 1970). The findings of this research are in line with the study of call centre agents and their oppressive work environments by (D'Cruz & Noronha, 2015). They report that the management's tendency to emphasise organizational benefits over employee interests is likely to exacerbate employee exploitation. As observed in Case 2 of this research, such exploitation results when management seeks employee compliance and commitment for their organizational imperatives to ensure organizational success.

Power asymmetry also posed some challenges for negotiation (on equal terms) in Case 1 as well. In this case the vendor BA had an advantage over his counterpart client BAs due to his powerful position as he was the vendor principal, COTS supplier and had extensive product knowledge as well as domain expertise. These factors potentially contributed to the power asymmetry between the client and the vendor that resulted in vendor dominance during negotiations. These findings are in line with those reported by Lauesen (2005),

who reports that once the client has purchased a COTS package software and wants to extend it, the vendor gains a de-facto monopoly because only the vendor can customize and integrate the system. According to Lauesen (2005), some suppliers can actually exploit this situation and may demand unfair compensation of their work. Although in Case 1, the client team members never reported unfair price charges by the vendor they certainly felt dominated by Vendor BA while negotiating for the contractually agreed scope, cost and schedule. The client team members also complained about having to cut back and accommodate scope because of lack of flexibility shown by the vendor while negotiating contractually agreed scope.

6.1.3.2 Power Asymmetry and the Challenges for Change Management

The offshore development team depended on the proxy client for several resources (information, job, cultural knowledge and language skills) and hence were underpowered. The dependence was also evident because the offshore team did not control any *strategic contingencies* (inability to make business decisions and to deal with uncertainties). These two factors have been associated with the basic forms of dependence and hence power (Bachrach & Baratz, 1970), as observed in Case 1. The power asymmetry posed several challenges especially during negotiations for people involved in implementing and managing changes.

Due to this power asymmetry, saying 'no' to the proxy client for frequent informal change requests and modified schedule demands (forced overlapping hours) was not an easy option for the development team. The informal changes, considered 'insignificant' by the proxy client, meant extra effort and uncompensated late sittings for the development team members.

Since the informal change requests by the proxy client were mostly communicated and discussed only with the offshore development manager, this resulted in communication and awareness problems among the development and quality assurance testing personnel. Due to lack of information and awareness, informal changes were often reported as bugs during testing, only to be later realized that these were informally and internally requested by the proxy client. Resolution of such problems not only required rework but also created conflicts and confusion among team members. Lack of information flow regarding requirements change between the business analysts and testers also resulted in ambiguous and non-verifiable requirements. These findings are consistent with those reported by Bjarnason et al. (2011) who explain that lack of communication of changes in

requirements among requirements engineering and testers contributes to wasted effort and frustration or lack of motivation to work with requirements. Such lack of communication according to Bjarnason et al. (2011) often results in output quality issues and failing to meet client expectations. These results are also confirmed by Uusitalo et al. (2009), who argue that involving testers early in requirements related activities is among the best practices to avoid requirements verification related challenges.

While this model worked in Case 2 with the offshore development site in Pakistan due to the availability of cheap labour and lack of implementation of existing labour laws (Syed, 2008), it may not be feasible for other developed and industrialized countries e.g. in Europe USA and Canada as in Case 1 (Royle, 2005). These findings have not been discussed in the requirements change management literature and present new insights relevant to a COTS product customization in a GSD environment (Hussain, Buchan, & Clear, 2014).

6.1.3.3 Overreliance on the Powerful Proxy Client

In Case 2, face to face meetings with the client were only held by the vendor BA/proxy since the offshore development team members never visited the client site. Due to his domain expertise and language skills the proxy client also enjoyed a central role in requirements related virtual meetings. This centrality of the proxy client's role in the stakeholder network in cross-organization collaborations provided him the autonomy to exert functional influence both within organizations and across organizations. It also created an environment where the offshore team members looked up to the vendor BA as an expert whose opinion was highly regarded especially in understanding requirements and client needs.

As a result, team meetings between the proxy client and the offshore team members to gain shared understanding of the requirements were dominated by the proxy client's opinion. This over reliance by the development team on the proxy client's understanding of requirements more than once resulted in misinterpretation of the client needs.

On one particular occasion, this misinterpretation caused a prototype rejection by the client and resulted in almost complete rework of the prototype; causing demoralization in the development team. This misinterpretation could be partly associated with misunderstanding between the proxy client and the developers. Paasivaara (2008) has reported similar findings while studying implementation of Scrum practices in GSD environment. She indicated that developers sometimes have a completely different

picture of the required functionality in their head compared to what the product owner wants which can result in undesirable implementations. As reported in Case 2, requirements misunderstandings were only identified during prototype evaluation, Passivara (2008), similarly reports that in one of her studied companies the misunderstandings were noticed during the iteration demo.

The challenges due to the powerful proxy client discussed above are in contrast to the traditionally perceived positive role played by the boundary spanners in GSD projects as reported by (Manteli, van den Hooff, & van Vliet, 2014; Nguyen-Duc et al., 2014, and Nisar & Hameed 2004). For example, Yadav et al. (2009), report the positive role of boundary spanners in collaboration and the success of requirements analysis activity. The bridging or boundary spanning role played by the proxy client in Case 2 runs in contrast to the perceived positive view traditionally held in the GSD literature. The challenges and conflicts inherent in the role of a powerful onshore boundary spanner or proxy client reported in this research have not been extensively explored in GSD literature.

6.1.4 Challenges Due to Client-Led RE process

Requirements engineering activities in Case 1 were led by the client organization who had hired three BAs to assess the selected COTS product and identify customization requirements. These requirements were then analysed, modelled and (initially) specified locally before they were shared with the remote vendor. Customization requirements went through another external RE cycle, involving the vendor (with roughly the same activities discussed above) and almost every time resulted in changes in requirements, prioritization and scope. Client-led requirements engineering, especially for COTS-based products, is not considered typical. Jarke et al. (2010), have argued that the industry has moved from bespoke and in-house software development towards purchasing COTS software; with a consequent trend towards 'vendor-led' requirements.

Involvement of client BAs in Case 1 to identify customization requirements, can be considered in line with a traditional systems analysis approach whereby the experts try to *'discover'* pre-existing requirements and then specify them, as discussed by Avison and Fitzgerald (2003). Bednar and Welch (2009) have argued against the presuppositions inherent in a traditional analysis approach and the notion of pre-existing requirements, and have suggested that system users need to take ownership and control of the analysis process. Although the involvement of client stakeholders in the analysis process is welcomed and advocated by participatory or client-led approaches for analysis, e.g. by Mumford (1995) and Checkland (1999), the involvement of external experts can

decontextualize the nature of inquiry, as emphasised by Bednar and Welch (2009) and as noted in Case 1.

Furthermore, the client BAs hired as external contractors in Case 1, were not domain experts and they had limited product knowledge. The additional layer of eliciting, interpreting and specifying user requirements (including changes) and communicating them effectively to the vendor BA was therefore a major challenge for the client team members.

In this situation where the client BA were leading RE activities, the differences in expertise, terminologies, and practices often resulted in friction between the vendor and client team members. Carrying out requirements and change-related activities over distance in the absence of a good vendor relationship and with competing roles across sites proved to be problematic for all involved. For instance, a professional rivalry and animosity emerged among the client and vendor BAs who complained about each other's lack of competencies, commitment and professionalism. Conflicts, deteriorating relationships and a lack of trust between the vendor and client team members ultimately led to resignations of two client BAs. These outcomes resonate with those reported by Morisio et al. (2000), who noted that having a close relationship with the vendor is one of the most common risk mitigation strategies in COTS based development and customization projects. These challenges are also confirmatory to the findings reported by Damian and Zowghi (2003), who report trust and good relationships is a must for effective client-vendor negotiations to achieve common grounds over requirements.

Another reason for the tensions and requirements related problems between the client and vendor BAs was the difference in their interests. The BAs and product owner at the client organization desired a unique business solution (by customizing the product to fit their organization's data, process and user requirements) whereas the COTS vendor personnel preferred a generic solution with minimal customization. Similar challenges have been reported by Hong and Kim (2002), who have explored conflicting interests from an *organization-fit* perspective regarding software packages such as ERP. They commented on the results by Swan, Newell, and Robertson (1999), and suggested that organizational misfits exist due to the conflict of interests between the product vendor and user organizations.

The implicit assumption from both the client and the vendor side was that the project would require around 80% adoption and 20% customization; in line with the general recommendation to opt for a packaged software (Kien & Soh, 2003). Upon

implementation however, the selected COTS product which was originally designed for the US market, required several significant country, sector, industry and organization-specific modifications. These software intensive customizations to the package however were too costly and disruptive for the project to be completed within the given budget and time for the project studied in Case 1. These findings align with those of Kien and Soh (2003) who report that having to customize package software is a common problem due to the feature misalignment of the package with the buying organization's needs. They assert that often clients don't really know the package software well enough to appreciate the implications of its adoption. They report results from a case study similar to Case 1, where an ERP system was customized to suit a) country b) sector, c) industry and d) organizational context of three Asian hospitals. They report that misfits arising from, country, sector and industry specificity) were more pervasive. These misfits were out of the organizations' control and required intensive customization that eventually outweighed the expected (cost and time saving) benefits of package customization for their case study as observed in Case 1 of this research.

On the other hand, in Case 2 (Pakistan and the USA) the vendor BA, who was also a domain expert, was responsible for leading requirements-related activities. There were no cross-site roles apart from a team of experts that was contracted to aid interpretation of user requirements. The development team utilized their expertise of the consultants whenever considered necessary but in doing so they did not report any conflicts. This reinforces the view of Jarke et al. (2010), discussed above advocating that a vendor-led requirements engineering approach should be taken in product customization environments.

6.1.5 High Level of Customization Requirements

At the time of contract signoff for the project in Case 1, many requirements and desired customizations were stated in terms of high-level goals and objectives. These high-level requirements (HLRs) and customizations were agreed upon initially, but the degree of modifications required to the software was inevitably unclear and negotiable. This meant that the project required flexible planning, scope management and a change-embracing RCM process since requirements elaboration would give rise to changes that would need reprioritization and negotiation.

The high-level nature of the requirements in the contract also made the separation of changes from requirements very difficult. On many occasions these HLRs and any related assumptions had to be clarified through synchronous and asynchronous communications

between the client and vendor teams. As a result, changes in requirements were frequently re-negotiated and the project scope had to be readjusted accordingly, through formal or informal change management processes. These activities required additional time and effort to be invested – or expended – in planned activities of the project. These findings are in line with those reported by Knauss et al. (2014), that when requirements evolution deviates from a typical and expected course i.e., from initial ideas getting clarified and implemented as stable requirements by the collaboration between stakeholders it usually linked to poorly understood requirements needing further negotiation or better alignment with project goals Knauss (2014).

In case 1, collaboration over requirements, scope management, prioritization and negotiation were not easy because of the poor client-vendor relationship and the constrained communication opportunities due to distance. These findings are consistent with those of Prokop (2014), who reported that, while it is not uncommon to see requirements for large and complex COTS-based development expressed at a high level early in the project lifecycle, project planning under uncertainty and efficient resource allocation to aid system design becomes challenging.

In the studied case 1, the challenges occurred in spite of the fact that the risks associated with requirements uncertainty in COTS development have been well established in the literature. Boehm and Abts (1999), warned the buyers of COTS packages about making blanket assumptions regarding the system requirements in the face of uncertainty. They also recommend not to use traditional processes for COTS integration and customization. As per Boehm and Abts (1999), specific risk-driven approaches should be applied due to the 'vagaries' of requirements in COTS-based integration projects.

For the WIPS project in Case 2, the contract also included high-level requirements, but because the project involved a *'functional rewrite'* of an existing web-based product (albeit with an improved GUI) there were fewer challenges encountered in requirements elaboration and verification. The existing website in Case 2 acted as a baseline for requirements that simplified collaboration over requirements and suggested changes.

6.1.6 Inadequate Vendor Expertise Onshore

The vendor BA was onshore with the client in the USA (Case 2) and acted as a 'boundary spanner' or 'intermediary' (Gregory, Beck, & Prifling, 2009) between the client and the vendor teams. The vendor BA's onshore presence meant that the client could engage in face-to-face communication with a competent conversation partner to discuss critical business and functional issues. The vendor BA facilitated knowledge transfer which was

made relatively easy due to the vendor BA's prior experience, domain knowledge and their frequent interpersonal contact with the client. This onshore vendor personnel presence and the availability of expertise in Case 2 were found to be helpful in requirements and change management activities particularly requirements knowledge transfer and requirements negotiations with the client. These findings are in line with contemporary knowledge in outsourcing projects as reported, for example, by Srikanth and Puranam (2014) and also by Keith, Demirkan, and Goul (2009) as well as Deshpande, Beecham, and Richardson (2011), who discuss the role and advantages of having onshore vendor personnel.

In contrast, the project studied in Case 1 had zero onshore presence of vendor personnel at the client site in New Zealand that resulted in a wide communication gap and difficulty in knowledge transfer. In contrast to the norm of having some vendor representation onsite with the client the zero onshore presence as found in Case 1, is atypical for outsourcing projects and is exceedingly rare as reported by (Srikanth & Puranam, 2014). This situation created challenges for achieving shared understanding of requirements, change negotiations and scope management. These findings again reinforce the value of having adequate onshore vendor personnel presence (around 20% to 30% of total team size) at the client location as reported by Venkatesh and Krishna (2005).

As experienced in Case 2, awareness of and timely access to project expertise was crucial to enable teams to foresee task issues accurately, coordinate more efficiently and contribute to project success. Onshore presence of the proxy client in Case 2 resulted in avoiding delays and complexities in resolving issues with the actual client and carrying out requirements-related activities. These findings are in line with previous reported results in various GSD studies (Clear, Raza, & MacDonell, 2013; Damian et al., 2003; Lanubile et al., 2013; Sarma & Van Der Hoek, 2006).

The findings of this research thus generally conform to the results reported in Herbsleb et al., (2005) regarding lack of domain and product knowledge being one of the key challenges in GSD projects. However, one particular finding stands in contrast to the failure experience reported by Herbsleb even with presence of onshore vendor representation. The presence of an empowered vendor representative (proxy client) as an onshore expert in Case 2, resulted in effective communication and coordination which led to relatively fewer client related difficulties for requirements and change-related activities.

6.1.7 Heavy Documentation

Heavy documentation and the effort to keep the documentation updated was considered a big challenge in Case 2. The client and vendor teams were geographically distributed and had limited opportunities for face to face communication except for the proxy client who was onshore with the actual client in the USA. In this situation the cross-site team members had to rely on formal communication mechanisms and explicit documentation especially for requirements and change related activities. These findings are consistent with those discussed by Fowler (2006), Razzak and Mite (2015), Thurimella and Maalej (2013) and Venkatesh and Krishna (2005), who suggest that explicit documentation is often required in GSD projects due to additional need for sharing knowledge because the channels for communication are constrained. Sharing explicit (documented) knowledge was however costly and demanded investment of significant effort. This however should not be a surprise for GSD practitioners according to Fowler (2006), who considers it as part of the price to do things offshore.

In Case 2, the involvement of a governmental client and the need for CMMI compliance resulted in a higher need for project related documentation. Furthermore, continuous changes in requirements placed additional burden to create and maintain the required documents. This effort overhead compelled team members to use shortcuts or skip some of the documentation altogether which resulted in missing some essential documents related to requirements such as use cases and test cases.

The absence of essential documentation meant that the team members could not share and exchange requirements or change related information in a timely manner. This gap in the knowledge transfer caused confusion and even conflicts between the development and testing teams and resulted in problems during testing and deployment. These issues are also reported by Sabaliauskaite et al. (2010), who suggest that skipping important documents often results in problems for downstream implementation and testing activities. Similarly, Rubin and Rubin (2011), report that when requirements related documentation is compromised, important knowledge may get lost during and after system development. These findings are also in line with the observations made by Fowler (2006), who suggests that heavy documentation not only requires people to invest time in writing them but also understanding things from written documents which is not always easy and often causes frustration (Fowler, 2006). Ideally, teams working in GSD should identify and strike the right balance between important and unnecessary

documentation however it is not trivial (Ågerfalk, Fitzgerald, & In, 2006) as observed in Case 2.

In Case 1, the client was leading the RE process and a significant amount of documentation was required for reporting purposes to the steering committee members and other stakeholders. In addition to this, the misalignment between the client and vendor's preferences of spreadsheet as collaboration technology forced the client to work to produce requirements and change related artefacts using spreadsheets rather than using JIRA. This placed additional burden of extra documentation effort on the client team and caused frustration for them as per the findings by Fowler (2006), reported previously. The client continued using JIRA and Confluence as their preferred collaboration technology however they faced challenges in keeping the separate data pools of specification templates, bug report and requirements changes using spreadsheet synchronised with JIRA and Confluence.

6.1.8 Inadequate Requirements Change Management Process

Change management practices in both studied cases (1 and 2) were found to be inadequate. In Case 1 new requirements as well as changes in the existing ones continued to pour in until the end of the project. Since the change management process in Case 1 was invoked only for high impact changes (i.e., changes equal to 10% or higher in project cost), the smaller and low impact changes were continuously entertained and accommodated in the project. This undue process flexibility resulted in an overwhelming number of unmanageable changes which affected the cost and schedule of the whole project. The initial project plan become unrealistic and had to be constantly updated which resulted in delays in planned release dates, additional project cost and additional time required to finish the project. A change rate of 30 % was experienced in the studied project for Case 1, which was at least three times more than the recommended rate of change for a project (Davis, 2013). Although Davis (2013, p. 176) does not consider change in requirements as 'bad' and recommends not to retard the flow of requirements change however he strongly advocates learning to manage the flow of requirements change. Given the globally distributed nature of the project and the inability of the client and vendor to *"meet regularly to decide intelligently which requirements will and will not move into the next software release"* it became a case where change management brought the project team to its knees as foretold by Davis (2013).

Similarly, Wiegers & Beatty (2013, p. 473) also view requirements change positively and suggest that some changes are not only legitimate they are unavoidable and even

advantageous. However, in case of scope creep, as seen in Case 1, where the project kept adding more functionality without any adjustment in resources, schedule or quality is insidious. Late changes in requirements as seen in Case 1 caused a heavy impact on cost and schedule as well as on the stability of the existing system. Wiegers et al (2013, p.473) also report that late changes may lead to frustration among the stakeholders regarding project completion time and delivery and often such projects do not get delivered at all.

In case 1, inadequate planning and anticipation of requirements change frequency led to the application of an unsuitable change management process. Following an unrealistic and impractical requirements management process in Case 1, allowed frequent scope creep and resulted in change management issues that are already difficult to manage in GSD environments (Lai & Ali, 2013; Šmite, 2006).

In Case 2, a CMMI based change management process was in place to manage changes in a formal way. However, this process was not followed internally by the Vendor BA/ proxy client who frequently made change requests throughout the project. These internal change requests were accommodated within the same project time and scope and without invoking the prescribed RCM process activities. This kind of scope creep would have resulted in many unwanted outcomes for the project such as missed deadlines, blown up project cost and schedule as pointed out by (Davis, 2013 and also Leffingwell & Widrig, 2003) however the development team bore its implications of having to work unrewarded extra hours and kept the project on course for on-time delivery.

As per Wiegers and Beatty (2013, p. 473) implementing requirements change is not free. A good strategy to cope with the 'forced' and unwanted informal changes would be to say 'no' as suggested by (Weinberg, 1995) however, the powerful role of the proxy client did not allow the development team members the ability to say no. For fear of getting reprimanded by the proxy client (CEO of the company) or even worse losing their jobs, the team members did not say 'no' to these informal requirements. Wiegers and Beatty (2013, p. 473) also note that "people don't like to say "no, "and development teams can receive intense pressure to always say "yes".

The offshore development team hence could not say no to the requests of the proxy client. Most of these informal changes in requirements were treated as 'emergency fixes' which had to be carried out immediately; hence leaving out almost all formal change management steps. Harjani and Queille (1992), consider the aforementioned fixes as *variants* of the formal or instantiated process that are often used to prevent a disaster or to modify software urgently. Time constraints on these changes make them incompatible

for a formal process of maintenance hence a short procedure becomes necessary. They report that, when these emergency changes are deemed necessary (which was almost always the case in Case 2) only a minimal solution and impact analysis is carried out by the most experienced development staff which is followed by change implementation and testing.

Somerville (2013, p.239), notes the issues with implementing changes quickly and directly into the system without following a formal change management process. He suggests that since changes are made directly to programs without modifying the requirements or design; the design and code become inconsistent. Furthermore, in situations where a quick and workable solution is chosen instead of the best solution, it accelerates software ageing.

In Case 2, informal changes were often discussed only among the proxy client and the development management. It created a problem of communicating change information to all relevant stakeholders such as the testing team. The lack of communication regarding informal changes created several other challenges for managing changes. These challenges included issues in baselining requirements, aligning requirements artefacts, managing traceability information, and verification. Rework and confusion resulted between developers and testers especially when the implemented changes in requirements were reported as bugs. These findings are in line with those reported by Sabaliauskaite et al., (2010) who note that requirements engineers do not always inform developers and testers when changes in requirements happen. It becomes extremely difficult for a testing team during the testing cycle to identify the right people who have change related information or developers who have implemented the changes. Since change related information is not updated testers are not aware about the changes that have occurred which makes not only testing the release problematic it also causes traceability and requirements verification to be a big issue as experienced in Case 2.

6.2 Summary of GSD Challenges

The model presented in Figure 6.1 summarises the new challenges, identified in this research, that particularly affect global RCM activities. The model in Figure 6.1 reflects the contemporary wisdom about known RE challenges reported in GSD literature that were experienced in the cases studied and reports new challenges that span the lifecycle activities of a requirement change. The model itself is an adaptation of the model presented by Damian and Zowghi (2003). The extension of the original model by Damian

and Zowghi (2003) was carried out by adding two main challenge categories (i.e. geographical distance and organizational diversity) and by further adding challenges identified in several studies (Lanubile, Calefato, & Ebert, 2013; Nurdiani et al., 2011, Noll et al., 2010; Šmite, 2006). The adapted model in Figure 6.1 confirms all challenges reported by Damian & Zowghi and some other challenges already reported in more recent literature for example, Lanubile, Calefato, & Ebert, 2013; Nurdiani et al., 2011, Noll et al., 2010. Furthermore, the adapted model presented in Figure 6.1 now includes three more activities, namely, *Identification of Changes, Management of Changes* and *Traceability,* to expand the model's coverage for requirements change management activities.

The key known GSD challenges experienced in this research comprised Organizational Diversity, Socio-Cultural Diversity, Remote Communication, Knowledge Management, Time Difference and Geographical Distance which influenced other GSD identified issues in this research (See Figure 6.1). Similar findings regarding these challenges and their impact on other GSD issues have been reported in extant GSD literature such as (Damian & Zowghi, 2002; Noll et al., 2010; Nurdiani et al., 2011; Šmite, 2006). The new challenges identified in this research impacting requirements change related activities were: 1) Immaturity of the Product, 2) Lack of product Knowledge, 3) Powerful Proxy Client, Client -Led RE process, 4) Inadequate Vendor Expertise Onshore, 5) High Level of Customization Requirements, 6) Absence of Onshore Vendor Expertise, 7) Heavy Documentation, and 8) Inadequate Change Management Process.

The impact of each of these changes on requirements change related activities in each case is noted in Figure 6.1. The challenges are considered new because they 1) are in addition to those identified by Zowghi and Damian (2002) and other studies on RE challenges such as (Lanubile, Calefato, & Ebert, 2013; Nurdiani et al., 2011, Noll et al., 2010; Šmite, 2006), 2) are in the context of this study (product customization in GSD) and 3) have impacted RCM activities in contrast to RE activities as reported by Damian and Zowghi (2002), 4) to my knowledge have not been previously covered in GSD literature. A possible modification to the model depicted in Figure 6.1 could be to remove the impact of challenges on individual case RCM activities and display all challenges (both known and the newly identified) as a summary that could be cited in future research in RCM and GSD area. Other than avoiding repetition, the main reason of not doing that was to allow readers to visualize the impact of newly identified challenges on individual cases based on their context. Besides, the identified challenges (both old and new) can

still be cited as issues that affect RCM in GSD without referring to the cases studied.

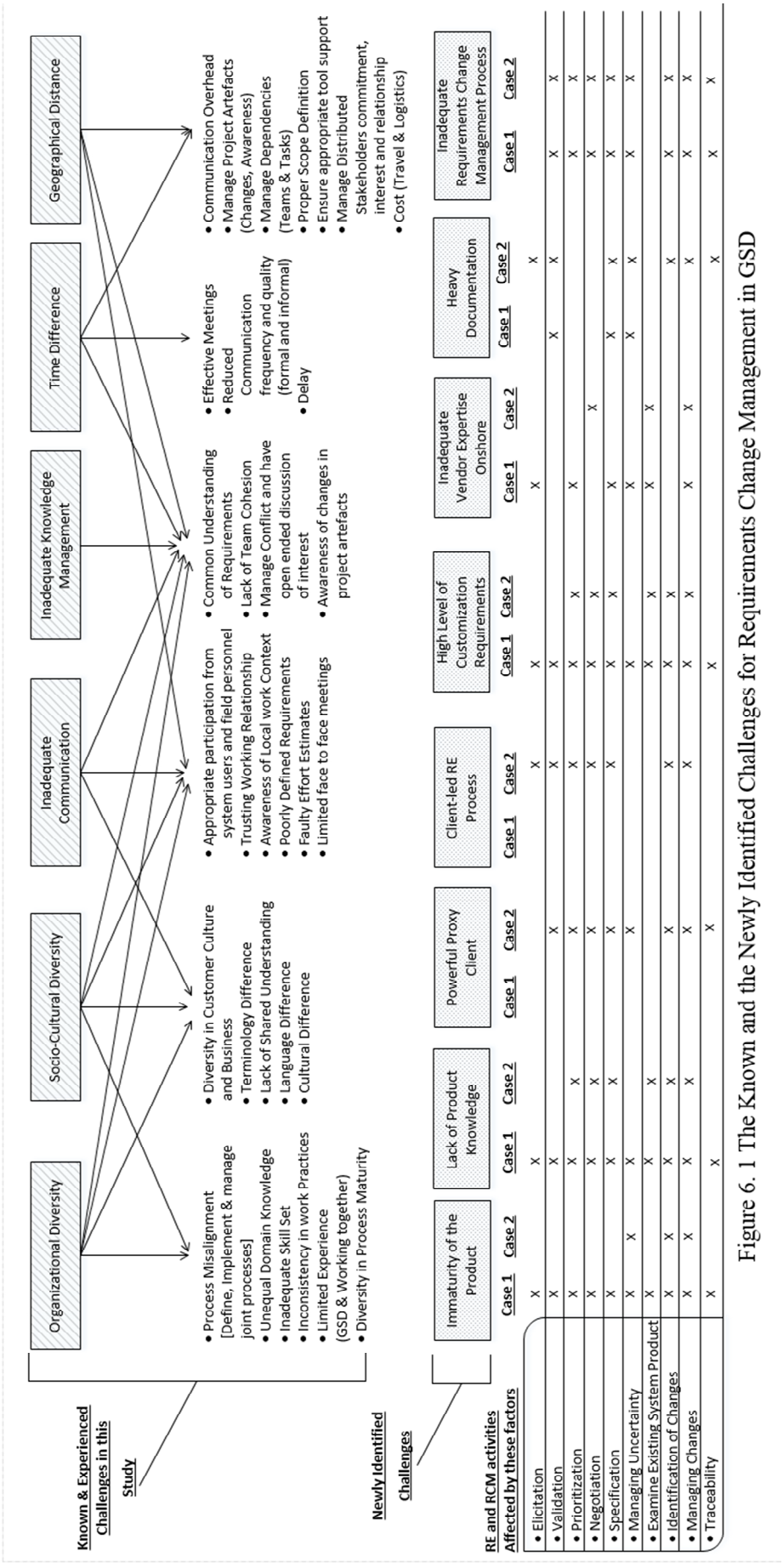

Figure 6. 1 The Known and the Newly Identified Challenges for Requirements Change Management in GSD

The combination of these newly identified challenges that made RCM activities difficult to perform for Case 1 was: High level of customization requirements, Immaturity of the product, Lack of Product Knowledge, Inadequate Vendor Expertise onshore, Client Led RE and inappropriate RCM. For Case 2 the combination that exacerbated change management process challenges and made it more difficult was 1) High Level of Customization Requirements. Powerful Proxy client, Heavy Documentation and Inadequate RCM process.

The impact of these newly identified challenges on various RCM activities (marked as 'x' under individual case) is noted at the bottom of Figure 6.1 depending on whether or not these challenges were experienced in each case

It is clear from the above classification that most of the challenges were people and process related. The newly identified challenges can be categorized under the three elements of a system namely people, process and product (Meso & Jain, 2006), see Figure 6.2.

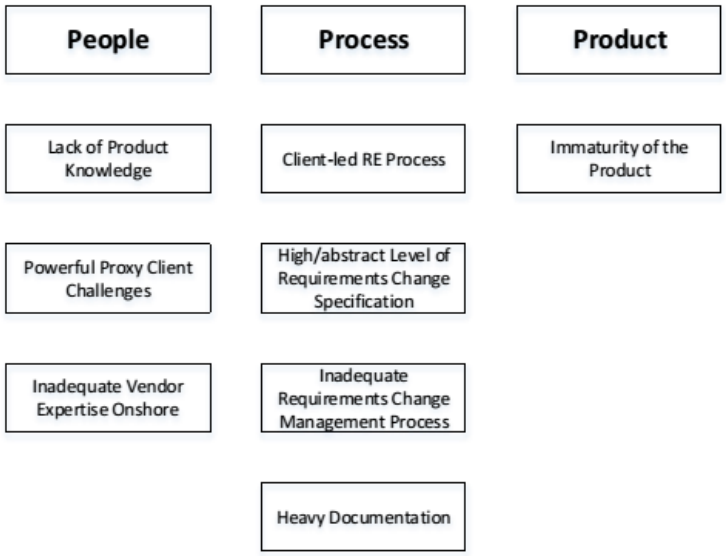

Figure 6. 2 Classification of newly identified challenges under three important aspects of software development: People, Process and Product

6.3 Role of Collaborative Technologies in RCM

Collaborative technologies (CTs) were considered a helpful and necessary means of collaboration and managing changes in requirements by all research participants. This perceived usefulness of CTs in distributed collaboration over requirements is consistent with the contemporary wisdom suggesting that the use of adequate CT reduces the impact of distance and supports requirements management in GSD reported in several studies

(Lanubile et al., 2010; Sinha et al., 2006b; Tell & Babar, 2012b). Use of CTs in general allowed distributed team members in both cases to clarify, record and disseminate change related information, reduce ambiguities, and enable knowledge and requirements change management. These findings are confirmatory to those reported by Sinha et al. (2006) and also by Sommerville (2011), who similarly deem CTs to be useful and essential in GSD especially for requirements management. Sinha et al. (2006), further assert that the use of CTs reduces impact of distance, helps in alleviating many GSD challenges and enables practitioners to carry out and improve distributed RCM activities.

However, CTs were not considered the main driver for project success in this research. The research participants considered the main factors for the success or failure of GSD projects to be good client-vendor relationship, leadership and management of people (the 'soft' side). CTs were placed at the second highest level of importance. This is in line with the findings from Cockburn's (2003) work where he states that "People are a first-order driver of a project's trajectory". Similarly, Damian (2007), has reported that overcoming people related issues in distributed stakeholder collaboration holds the key to successfully managing requirements in GSD.

The findings reported in this case study are mainly confirmatory in terms of the supportive role of CTs in managing requirements. The participants valued support for code management, requirements management and team collaboration provided by their modern collaborative development environments (CDEs) such as TFS in Case 2 and Confluence & JIRA in Case 1 as discussed and desired by Booch and Brown (2003).

CTs were also reported to cause some hindrance in carrying out some of the activities related to requirements and change management. In Case 1, the vendor organization did not adopt project wiki (*Confluence*) that created knowledge gaps between the client and vendor organizations especially for business analysts and developers. Similarly, some of the key stakeholders found the interface complexity and learnability of the *Confluence and JIRA* to be very challenging and therefore did not use them for information sharing or retrieval. This again created problems with regards to change related knowledge sharing. These are confirmatory findings to those reported by (Karhu, 2011) who proposed that improved wiki usability may lead to users embracing the wiki way of working and increased productivity whereas the continued use of a wiki with low usability leads to productivity loss. Similar results are reported in a survey of Wiki users by Hester (2010) whose findings indicate that the ease of use is one of the most important factors that positively influence the use of wiki technology (Kiniti & Standing, 2013).

However, the participants also encountered problems while working with issue tracking software such as a) timely retrieval of accurate change/ issue information either due to unavailability of information or preference of using other tools for managing change or issue related information. These findings are in line with the results from the Halverson, Ellis, Danis, and Kellogg (2006) study reporting many challenges ranging from identifying relevant information about particular issues, resolving interdependencies in task allocation (in order to avoid stepping on other people's toes), retrieval of historical information and accurate reporting of issues to an appropriate level.

Participants in both studied cases testified the usefulness of collaboration support provided for issue tracking tools in combination with a collaborative development environment. They considered them as a) the key for managing change related activities, b) central to prioritization and coordination c) vital for tracking state of the project and people involved and d) crucial for providing a historical record of a team's activity as development progresses.

Similarly, Email was generally considered hugely beneficial as an asynchronous communication technology. However, emails messages were also reported to have been misinterpreted on a number of occasions. Cross site communications through emails were always vulnerable to misunderstandings and misinterpretations because facial expressions, tone of voice and other (contextual) views are not available to the recipients. The participants considered that even though they were careful in writing email messages there was always a possibility of misinterpretation because the people involved belonged to different organizational and national cultures and backgrounds. These findings are similar to those reported by Friedman & Currall (2003), who assert that communications through email leads to misinterpretation regardless of how carefully the messages are crafted. Similarly, O' Sullivan and Flanagin (2003), point out that email messages can be misunderstood by other people just because they do not belong to the same organizational culture. They argue that misinterpretation happens because email messages are often understood within the framework and specific norms of the organizational culture.

Emails were not always the first choice for cross site communication in fact they were avoided in resolving cross-site negotiations and conflicts especially in Case 1. In conflict management, emails were seen as tools that introduced delays and lacked support to resolve conflicts therefore the participants preferred to use other rich media for communication such as web or video conference. These are confirmatory findings as

reported by Friedman and Currall (2003), who noted that email should not be used as a medium to settle disputes or engage in arguments with another party as it can exacerbate conflicts. The findings regarding emails as CTs in this research are similar to those reported by Damian & Zowghi (2003), who suggest that emails alone are not suitable for distributed stakeholder requirements negotiations rather they should be supported by rich media in order to reach common grounds.

Web conferencing tools were again reported being mainly useful in providing collaboration support for requirements and change related activities. However, connection problems, lack of access rights to a use a particular tool, poor audio quality and echo caused misunderstanding and frustration on numerous occasions. Similar challenges are reported in GSD collaboration for requirements by Niinimäki et al. (2010) as well as Jaanu, Paasivaara, and Lassenius (2012). Both studies in GSD projects concur that low sound quality was the main challenge in audio conferencing which made it difficult to understand what was being said.

Spreadsheets as collaborative technologies were identified to have played a crucial role supporting distributed stakeholder collaboration over requirements in both cases studied in this research (discussed in Section 6.3.2). However, there were also some issues in working with spreadsheets such as having an inconsistent view of requirements, information fragmentation across several media causing lack of shared understanding, having to update various versions of each document and generating isolated and contradictory data pools. These findings are generally consistent with the case study observations by Hanisch and Corbitt (2007) in their study to understand RE challenges for GSD. They report that the multiplicity of contributors to Excel spreadsheets and information splitting across many spreadsheets resulted in difficulties such as delay in querying and reporting information from these sources. Table 6. 1 presents the collaborative technologies used in the studied projects and their helpful as well as hindering role for collaboration support during RCM process.

Collaborative Technologies	**Helpful Role**	**Hindering Role**
Project Wiki and Bug Tracking Software	Store, track and manage requirements/change and test related artefacts Support traceability, record change identification information, managing changes and bugs, status reporting Support collaboration	Tool complexity and high volume of Information Learnability Challenges (e.g. Complex Interface) Disfavored Collaborative Technology as compared to Spreadsheet

	Support activity awareness and Knowledge Sharing	
Email	Preferred tool for asynchronous Communication, Collaboration and Knowledge Sharing Vehicles for carrying requirements, change and test related artefacts	Misinterpretation of messages Not suitable for conflict resolution Not suitable for negotiations and conflicts
Skype and GoToMeetings	Synchronous communication, Collaboration Allowing Knowledge Sharing through Discussions Requirements and scope Negotiations Requirements clarifications and conflict resolution Requirements Prioritization Supportive in finalizing specification Supportive in requirements handover activity Enablers for collaboration in GSD environments	Misunderstandings and frustrations due to poor audio quality Latency and echo causing confusion and frustration
Spreadsheets	Support requirements storages and sharing Facilitate knowledge sharing and collaboration as containers of requirements/change related artefacts (workflows, annotated diagrams, product modification images) Aid requirements and change review, negotiation, specification, analyses Facilitate bug reporting and resolution (through discussions/comments) Facilitate cross site RCM practice evolution	Inconsistent view of requirements Information fragmentation across several media causing lack of shared understanding Keeping various versions of each document updated Generating isolated and contradictory data pools

Table 6. 1 Collaborative Technologies Used in the Studied Projects with their Helpful and Hindering Role

6.3.1 Efficacy of Existing Collaborative Technology Repertoire

The repertoire of CTs used in both the studied projects was quite similar across projects for in both studied cases emails, bug tracking tools, spreadsheets, and web conference tools were used for collaboration over distance. While answering questions based on Activity Based Computing's core principles (Tell & Babar, 2011) about their existing CT infrastructure and its efficacy to manage requirements change activities, the research participants were quite content with the CTs in place. Except for the occasional challenges with CTs faced by the participants (perceived as 'routine'), they were satisfied with the support provided by existing collaboration technologies for communication, knowledge transfer, information and artefact sharing, collaboration and awareness in their GSD projects.

These findings are contradictory to those proposed by Tell and Babar (2011, 2012) who argue that existing GSE tools cannot provide adequate support for collaboration because

of the desktop metaphor that underpins these tools. They asserted that a unified platform approach needs to be utilized to remove limitations of existing tools to improve information needs of communication, collaboration, coordination and awareness among distributed team members Tell and Babar (2011, 2012). They recommended that a new Activity Based Theory paradigm was needed to be introduced to address challenges of tools in GSE. However, the participants in both projects argued that additional tools were not needed since the existing collaborative technologies in place adequately fulfilled their distributed work activity needs for communication, awareness, knowledge management, control and coordination. The participants did not want their CTs to be provided under a single platform and were not concerned about the additional effort in challenges such as context switching (Tell and Babar 2011; Tell and Babar 2012). The findings in this research are confirmatory to those reported by Bjorn et al., (2014) who conclude that people in distributed software development are comfortable with the existing collaborative technologies and have already made them part of their work. They assert that collaboration technology readiness (difficulties faced in adapting, adopting, and bringing collaboration technologies into use (Olson & Olson, 2000)) is no longer a pertinent challenge for collaboration across distance of CSCW systems. The responses received from the participants of this research are similar to the findings reported by Bjorn et al. (2014), where the participants also clearly expressed that they had no problems for which they required new technology as a possible solution and a new technology is the last thing they needed. These findings however are contradictory to the recommendations reported by (Tell & Babar, 2011) who argued that exiting CTs in GSD are not adequate for distributed collaboration and do not prove the desired collaboration support (based on the six core principles of ABC).

The offshore development team did value the collaboration support provided by team foundation server (TFS) for code management, communication, however, some of the more technically savvy participants (from Case 2) actually advocated for the use of '*dedicated tools for dedicated tasks'* approach over a heavyweight generic tool provided for all development as collaboration needs. The approach to use dedicated tool support has previously been promoted by (Salger, Sauer, Engels, & Baumann, 2010) who suggest that a "dedicated tool support is key to success and supports a seamless transfer of knowledge between GSD teams". This is also a finding in contrast to the suggested approach provided by Tell and Babar (2011, 2012) where they recommend that CTs

should be united and put under one platform to adequately support distributed collaboration.

In Case 1, the reluctance to use a single platform to combine the existing set of CTs might also have been caused by the lack of awareness of such suite of tools (e.g. JAZZ Team Concert platform). When interviewed, the participants from Pakistan site were not aware of any such tools and asked the researcher to provide examples of such tool suites.

Furthermore, the participants favoured having dedicated tools for specific tasks due to the perceived complexity of a generic tool if that was to cater for all collaboration needs. It may also have been a resource challenge in terms of cost i.e., setting up a base platform and establishing and reinforcing the standards of technology use across sites. These findings are contradictory to those reported by Bjorn et al. (2014) who consider that practitioners in GSD have increased knowledge of the existence and usage of available collaborative technologies as these technologies have seen a rise in their use in the workplace. They argued that the more pressing issues have more to do with the instability of CTs rather than their lack of knowledge or expertise in their application or use which clearly contradicts with the findings of Case 1 in this research.

A closer analysis and mapping of the existing repertoire of CTs used in both cases studied revealed that these CTs could be mapped to three main process categories that require support by GSD tools (See Table 6.2) as recommended by Rodríguez, Vizcaíno, Ebert, and Piattini (2010). This further reinforces participant claims regarding the adequacy of existing technology. The existing CTs in both cases could be mapped to the three main process categories requiring support by GSD tools according to Rodríguez et al. (2010). These categories are a) *Project processes* (that cover planning, coordination and control activities),

b) *Implementation processes* (covering requirements, analysis, architectural design, Construction, Testing and Integration activities) and

c) *Software Support Processes* (that cover, Documentation management, Quality Assurance and Testing activities). Table 6. 2 shows that in both projects studied (Case 1 and Case 2) the existing collaborative technologies were available to support each of the three process categories requiring collaboration support.

GSD Tool Supporting Project Processes	GSD Tool Supporting Implementation Processes	GSD Tool Supporting Software Support Processes

(Planning, Coordination, Control)		(Requirements, Analysis, Architectural Design, Construction, Testing & Integration)		(Documentation, QA and Testing, Communication and Collaboration over distance)	
Case 1	**Case 2**	**Case 1**	**Case 2**	**Case 1**	**Case 2**
Ms Project	MS Project	Spreadsheets	Spreadsheets	Confluence	Spreadsheets, SVN
Confluence JIRA	TFS	MS Visio	MS Visio	JIRA	In-house Bug Reporting & Tracking Software
Email	Email		Visual Studio.NET, Team Foundation, Version Control (TFVC)	Email, Skype, GoTo Meetings	Selenium
			Selenium		Email, (In House) Chat Messenger, MS Lync,

Table 6. 2 Collaborative Technologies Supporting Three Collaboration Intensive Process Categories

In line with the results shared by Rodríguez et al. (2010) the collaborative tools in use as listed in Table 6. 2 were able to support and somewhat minimize (to a certain degree as reported by the participants) the effects of distance however none of these tools were able to reduce socio cultural distance. This is in contrast to the findings by Rodríguez et al. (2010) who report *ActiveCollab* and *Assembla*, were able to reduce even socio cultural distance by providing profile info and social networking.

The participating members were not 'fazed' by context switching considered as one of the main motivators for implementing collaborative work environments in distributed development settings reported by Tell and Babar (2012).

In the cases studied (Case 1 and 2) the organizations were working with a combination of collaboration technologies and groupware environments (Email, Web Conference tools, Wikis and Spreadsheets) and quite often (synchronous) cooperation between the client and the vendor was complemented by asynchronous sharing of documents as suggested by Prinz, Martinez-Carreras, and Pallot (2011).

6.3.2 Spreadsheets as Collaboration Technologies to Manage RCM Activities

Spreadsheets were found to be useful CTs by the research participants that facilitated almost all requirements management activities including analysis, prioritization, negotiation, specification, design, validation and traceability. Due to their flexible design and the ability to act as repositories for various requirements related artefacts they played a crucial role in cross-site requirements/change communication and collaboration.

Stakeholders engaged in collaboration activities using spreadsheets (in combination with communication technologies such as *MS Lync, Skype* and *GoToMeeting)* to visually contribute towards the final form of the shared requirement artefacts (e.g. requirements specifications or sprint backlogs). In contrast to the common perception of spreadsheets as single-user 'tools'; lacking support for coordination, communication, collaboration (Tell & Babar, 2012), they were seen as one of the most relied on collaborative technologies (along with email) for managing RCM activities. In this research, spreadsheets allowed the flow of work in fluid and informal ways across users that resulted in facilitating cooperation among their users as reported by Nardi and Miller (1991). Their flexibility and tailor-ability made them a tool of choice for the team members in both the cases studied.

These findings are consistent with those reported by Malone, Lai, and Fry (1995) who report that spreadsheets are "perhaps the best known examples of existing radically tailorable systems". Similarly, Nardi and Miller (1991) concluded that the translucent nature of spreadsheets with "surprisingly strong support for cooperative development of a wide variety of applications" make them a tool of choice for many organizations, as found in the studied organizations as seen in this research.

These findings regarding the use of spreadsheets for collaboration over requirements as well as issues identified in this research are generally consistent with those reported by Damian and Zowghi (2003) as well as Tell and Babar (2012), who state that GSD practitioners often adopt spreadsheets specifically for requirements management and issues/bug tracking. The extent of spreadsheet use and its coverage for requirements and change related activities in this research are much broader in scope and resemble the lifecycle approach (from vision to delivery) adopted by Ebert & De Man (2005), to identify causes of requirements uncertainty and project failure.

The use of a set of inter-related spreadsheets especially in Case 2 closely resembled the usage reported by (Ebert & De Man, 2005) who noted that in certain agile projects one multi-purpose spreadsheet was used to manage the entire project. Such spreadsheets, according to the authors were used to record "all requirements, their status, effort, responsibilities and mapping to increments, test cases and work products" as found in Case 2.

Similar to the findings in Case 1 and 2 regarding the use of spreadsheets, the requirements analysis and specification for NASA's *Orion* crew exploration vehicle spacecraft flight was supported by the use of spreadsheets as reported by Prokop (2014). In *Orion;* a COTS

based software development project, a master spreadsheet (of 3,166 requirements with estimated 1.1 million lines of code) was compiled for software requirements specifications (SRSs) from 17 different SRS documents which was then analysed, evaluated and used for SLOC estimation. Prokop reported that spreadsheet-supported requirements compilation and analysis aided in requirements understanding and analysis, effort estimation, software simplification, migration and implementation. These are similar findings to what have been found in this study and were reported in (Hussain, Buchan, & Clear, 2014).

6.3.3 Spreadsheets and Change Management Practice Evolution

The findings in this case study revealed that RCM practices, the associated artefacts and the collaborative technology that enabled coordination are all closely implicated and co-evolve in the process of local and global collaborations. For example, in Case 2, the flexible design and nature of spreadsheet based artefacts allowed their structure and content to be modified that resulted in the evolution of RCM practices. For example, cross-site team members started to use time-stamps and role signatures to clarify and improve communication.

The flexibility of spreadsheet to accommodate changes in practice made possible through the changes in its content and structure allowed a "lift in the professional behaviour at the vendor site, increased cross-site visibility of all changes and a global accommodation of new requirements negotiation practices" (Hussain & Clear, 2014). These findings are akin to the processes reported by Orlikowski (1996), who assert that organizational actors enact changes in practice as they adjust, innovate, and improvise their work routines over time. As seen in the modification and evolution of the Requirements Change Log (RCL) in case 2, Orlikowsk reports that subtle changes were enacted in the studied organization where "organizational actors appropriated the new technology into their work practices, and then experimented with local innovations, responded to unanticipated breakdowns and contingencies, initiated opportunistic shifts in structure and coordination mechanisms, and improvised various procedural, cognitive, and normative variations to accommodate their evolving use of the technology".

6.3.4 Collaborative Technologies and RE Process Alignment

The process differences inherent in client and vendor organizations led to difficulties with aligning cross site RE processes and keeping consistent tool usage to support RE. Tool appropriation and preferences prevented effective implementation of cross site practices for knowledge management and change management. These are similar findings to those

reported by Damian (2007), who report that aligning RE processes and tools support is challenging in GSD because remote stakeholders at different sites often follow different processes for requirements analysis, documentation, and change management.

Tool preferences in Case 1 caused inconsistency in RCM practices and caused conflicts as well as frustration for the client team members which affected client-vendor relationships. The preference of the vendor to use spreadsheets over bug tracking software (JIRA) in Case 1 resulted in challenges for the client team members. Similarly, in case 2 the vendor team preferred to use spreadsheets over bug tracking software (developed in-house) for requirement and change management activities. This aligns with the results reported by Monteiro, Ebert, and Recknagel (2009) where they noted that partner companies "usually don't work on the same requirements repository or use the same RE tools". These findings are also consistent with those reported by Lanubile et al. (2010), who assert that none of the existing tools or collaborative development environments fully support all the necessary activities for GSD therefore global practitioners appropriate communication and collaboration technologies in particular ways to assist requirements related collaboration, as seen in Case 2 and also reported by DeSanctis and Poole (1994).

6.3.5 Use of Rich Communication Media for RCM Activities

The remote vendor and distributed stakeholders meant greater separation between the problem owners (the client/business community) and problem solvers (Vendor and his development team). In the studied cases for this research, distributed collaboration suffered due to distance and created challenges for RE such as knowledge management, process alignment, and RCM. These challenges are confirmatory for GSD projects and are extensively reported in GSD literature Damian (2007), Herbsleb (2007), Smite (2007) and Nurdiani (2011).

In both cases studied the use of web conference tools such as Lync, Skype and GoToMeetings were effectively utilized to support RCM activities over distance. In particular web conferencing tools with screen sharing support were by far the most useful feature for collaboration over RCM artefacts. Cross site team collaboration using application sharing facilities of web conferencing tools (e.g. screen and document sharing) enabled team members to work on a shared document during an audio/video conferencing session. To ensure consistency of distributed but shared artefacts such as requirements or design specifications simple 'floor control' mechanisms were applied similar to those reported by Prinz et al. (2011), such as allowing only one user to edit the document (e.g. Vendor BA or Client BA in case 1) and then passing it on to other users

when they wanted to take over. This allowed frequent information sharing, evaluation and progress monitoring practice using commonly defined artefacts and also as a problem solving communication strategy. Thus in both the studied cases, supplementing a synchronous cooperation pattern with several asynchronous steps was used as an approach suggested in CSCW practices as reported by Prinz et al. (2011). These findings are also generally consistent with the recommended GSD collaboration practices reported by Paasivaara and Lassenius (2004), who report that successful projects ensured two-way communication regarding project progress, sharing artefacts across sites and establishing practices to build and maintain working relationships.

The findings related to the use of synchronous tools in combination with an asynchronous medium of communication (Emails) as well as requirements artefacts is a useful practice in cross site collaboration as seen in Case 1 and Case 2. These findings are also in line with the results reported by Smite (2006), Damian (2007) and Ebert and De Neve (2001) who report that small distributed teams have found video or teleconferencing in support of weekly meetings useful for requirements elicitation, validation, and management activities. In Case 1 however, these practices were unsuccessful in resolving issues related to conflicting roles between client and vendor teams.

The use of rich media such as video conference was not utilized by the participants in both cases. The participants reported to have used audio conferencing tools and their screen sharing features for requirements related collaboration. In case 1 the team members did not prefer to use this option whereas in the case 2 the team members said that they did not use it because it was not a project requirement. These findings are aligned with those reported by Lanubile (2009), who suggests that video conferencing is known to be problematic not only due to its the infrastructure cost and maintenance overhead but also for the difficulty of conducting long running discussions where more than a few people are involved.

In summary, collaborative technologies played a supportive role in requirements management activities. The problems caused due to the existing array of CTs did not pose threat to the project and were not of serious concern to the participants. The problems that caused major issues and even impacted the outcome of the projects were related to leadership, resource management, client vendor relationship and trust.

These findings are consistent with those reported by Damian (2007), she concludes that global software engineering work is challenging not on the technological front but on the people side. The problems related to leadership, project management and other pre-

contract upstream processes led to many downstream problems for RCM. These findings are consistent with those reported by Ebert (2005), who reports that lack of focus and attention on upstream processes (before the project start) and decoupling them from RE processes results in project deficiencies, for example managing changes in requirement and eventually delayed releases, as experienced in this case study. Collaborative technology was indeed used to support rectification of these issues but it was not the driver of the projects, furthermore it was neither the causer nor the eliminator of challenges, its role was seen more as a rectifier, supporter and facilitator.

6.4 Aggregated Model for Requirements Change Management

RCM activities in practice were analysed in detail for both the cases studied to identify the lifecycle of a requirements change in this research. The analysis enabled the researcher to produce a synthesized but abstracted version of the RCM process models from Case 1 and Case 2. The synthesized model depicts the distilled version of core and peripheral RCM practices identified as similar across the studied cases (See Figure 6.3). It draws upon mutually-supporting multiple data sources and methods which are: interview data, electronic artefacts, process mapping and close observations of practices based on significant periods in the field setting. This closeness to practice enabled the researcher to identify divergence between practitioners' actions and the both the prescribed models in the organizations as well as the classical models of requirements change management found in literature. The synthesized model depicted in figure 6.3 encompasses both the formal and informal change activities empirically derived and abstracted from the two cases studied, and therefore is titled the "Dual Formality RCM (DFRCM) model".

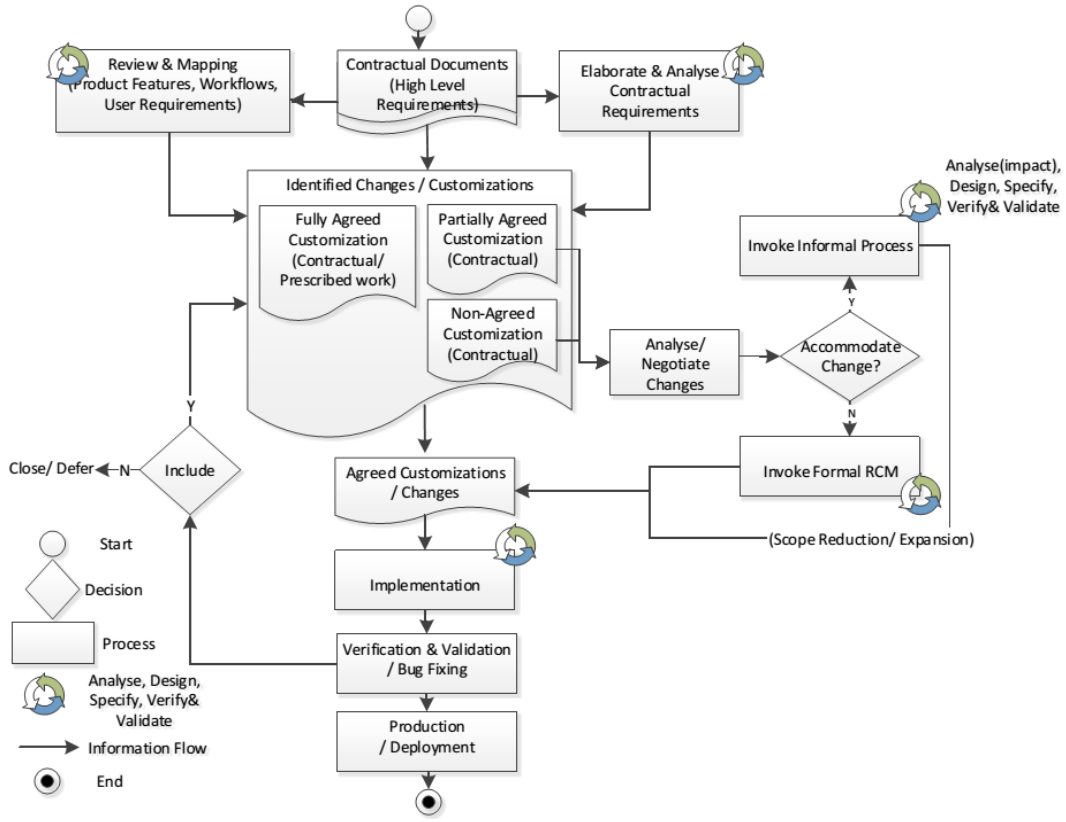

Figure 6. 3 Dual Formality Requirements Change Management (DFRCM) Model.

The model follows the commonly adopted requirements change management activities reported in the literature *(request, analyse, verify, implement, validate and update/deploy/close)* of the contemporary RCM models (Hussain & Clear, 2012; Imtiaz, Ikram, & Imtiaz, 2008; Leffingwell & Widrig, 2003; Niazi et al., 2008; Sommerville, 2011) . This model (Figure 6.3) however, can accommodate the deficiencies of an initial RE cycle (which may result in fully agreed, partically agreed and non-agreed requirements) and further adds the new dimensions of informal change and the activities for its treatment in both pre and post implementation. The novelty of the model also comes from the fact that it can be used both for traditional and iterative development methodologies. In case of Waterfall- like methodologies there would be only one 'go' or iteration from start to finish while for value driven methodologies (e.g. Scrum) there could be as many interactions as required for the project.

The model in Figure 6.3 shows that detailed requirements are elaborated from high level contractual requirements that potentially contain changes in requirements as well. These potential changes to requirements go through an analysis and negotiation process where they either follow a formal or informal process before being agreed upon and finalized for implementation. The bugs and comments from the users are identified and discussed before (formally or informally) getting reinstated in the next development cycle. The

model in Figure 6.3 at present is not specific for GSD however, it holds the potential to be extended to cover distributed development projects (discussed in Chapter 7, Section 7.7).

Existing requirements change management (RCM) models found in the literature are geared towards handling formal requirement changes only. They fall short in recognizing and dealing with the unofficial or informal requirements changes that are endemic to many software projects. Their underlying assumption is that a requirement can (and must) be classified as a change only after the requirements are base-lined or signed off. However, in reality the interplay of the dynamic and often uncontrollable sources of requirements change does not allow a change to be held back until the formal ceremonies of specification agreement have been carried out.

The Dual Formality RCM model in Figure 6.3 however, highlights the prevalence of InfRC in both studied cases in this research but may as well refer to both collocated and distributed software development projects in general. InfRC as an emergent theme identified in this research and depicted in this abstracted model (Figure 6.3), contrasts sharply with the typical notion of requirements change found in the literature where changes are generally perceived negatively and the concept of informal changes is virtually non-existent. This research asserts that InfRC could be further viewed as innate in all software development, although the challenges of accommodating InfRC became exacerbated in the global software development (GSD) environment.

This research illustrates with evidence, the dichotomy that exists between theory and practice in dealing with InfRC discussed in detail in Section 6.4.1. No matter how such changes may be managed we need to recognize them and prepare ourselves technically, emotionally and managerially to be able to address them as part of the change management process – or more generally as a necessary development activity.

6.4.1 Informal Requirements Changes and Reflections on RCM Models in the Literature

6.4.1.1 Inevitability of InfRC

A key finding from analysis of the actual change management processes in this research is the emergence of *Informal Requirements Change* (InfRC) (as noted in Figure 6.3). InfRCs were inevitable and pervasive in the cases studied for this research. The reasons for InfRC could have been the nature of software development itself where requirements may never be complete. Similarly, InfRC may also be considered a natural outcome of

software procurement and development projects where “open-target requirements” are recommended since it is hard to specify requirements in detail upfront (Lauesen, 2006). In such contexts the customers specify their demands and expectations and the vendors respond with how they can meet the demands. Again, this approach leaves plenty of room for requirements elaboration and change, some of which may end up being informal change requests.

6.4.1.2 Formality of RCM Models and InfRC

The focus on formal RCM that is prevalent in the literature has meant that InfRC has received very limited treatment (Davis, 2013; Leffingwell & Widrig, 2003; Sommerville, 2011; Wiegers & Beatty, 2013; Wiegers, 2000). Requirements change in general is often viewed negatively (Hussain & Clear, 2014). Handling of informal changes is similarly viewed as a negative activity infamously associated with “scope creep”, “Gold plating”, “creeping elegance” (Leffingwell & Widrig, 2003, p. 345; Wiegers & Beatty, 2013) and “skunkworks activity” (Bommer, DeLaPorte, & Higgins, 2002). For example, Wiegers (2000) considers inadequate process to deal with requirements changes as the most glaring symptom of requirements trap that has to be avoided. Wiegers (2000) reports on change related activities where people bypass the process to get their desired changes incorporated in the software in informal ways (for example directly asking the developers to make changes) but he categorizes such practices as deficiencies in process something that needs more ‘formalization’ (Wiegers, 2000). Leffingwell and Widrig (2003) similarly propose more formality for RCM process models with formally imposed control policies to plan, implement and manage changes through a formalized single channel and mechanism to control changes. In the classic requirements change management models (Niazi et al., 2008) formal processes driven by CCBs and CRFs is a typical approach which ignores informal processes for dealing with requirements change.

Approaches to its treatment therefore appear to have been either focused towards more formalization or overlooked altogether, apart from the agile movement which focuses on permitting and even encouraging change (Regnell & Brinkkemper, 2005). While agile processes address a change orientation they typically have some notion of a baseline, and apply practices of time-boxing and prioritization as the primary RCM mechanisms (Pichler, 2010). The literature indicates a contrast between the traditional and agile approaches, with the former focused on task and control and the latter on people and practices.

The next section evaluates the research carried out for this thesis.

6.5 Research Evaluation

In traditional requirements management literature, changes in requirements only occur after requirements signoff. Changes in requirements following the signoff are considered formal and are supposed to follow a formal change management process (aligned with the views initially held by the researcher).

A broader scope of requirements change lifecycle activities emerged, through careful investigation of the cases studied, in response to RQ1 (i.e. the challenges faced by global practitioners in RCM). The lifecycle of a change starts from the initial agreement on high level requirements and proceeds to the requirements signoff and right through to their implementation and testing. The initial agreement on high level requirements is reached either through RFP or other contractual documents in plan driven methodologies whereas in value driven methodologies the initially agreed vision and scope is captured in the product backlog. A change in this broader and more complete lifecycle is considered to go through both the formal and informal treatment.

This thesis represents the product of an ongoing and evolving process of learning. As the research developed, the focus broadened from the exploration of requirements lifecycle challenges only after sign off to include upstream and high level requirements elaborations and clarifications that resulted in requirements change. The research also focused on the associated role of collaborative technologies, towards understanding a more realistic and in-practice lifecycle activities of a change. The study was driven by literature, empirical findings and the previous global requirements change management model provided by Hussain & Clear (2012).

In respect of RQ2 (i.e. the role of collaborative technologies in RCM), at the start of the interviews the researcher expected to hear a desire expressed by practitioners for more collaborative technology, and also to have such technologies made available under a single platform. However, right from the first few interviews the practitioners expressed reluctance with regards to having more technology. Furthermore, they favoured dedicated tools rather than one suite of technology to support all their requirements and development related collaboration. The practitioners did not consider technology to be the panacea for their troubles. The researcher therefore focused more on understanding the role of individual collaborative technologies within the existing CT repertoire, for example spreadsheets and their support for RCM activities.

While talking about the collaboration support provided by the existing CTs, the interviewees could not identify any area where existing CTs were not able to support collaboration (based on the six desired areas of support according to activity based computing principles). Therefore, the questions to explore the role of CTs based on activity based computing's six main principles were not helpful and had to be modified. Furthermore, the interview questions related to critical incidents had to be slightly adjusted when the researcher realized that the practitioners were having difficulties in recalling incidents that critically affected their work or productivity at the organizational level. This was partly to do with interviewing as a technique where the participants had to recall incidents on the spot about technology which is often not noticed until a breakdown occurs as noted by Winograd and Flores (1986). The participants interviewed later were instead asked to think about critical incidents that impacted their own work or productivity and the role played by the collaborative technology.

As the study progressed the researcher also realized that the role of CTs was critical for the distributed collaboration requirements but CTs were taken for granted by the practitioners (Cajander, Daniels, Cullhed, Clear, McDermott, & Laxer, 2012) in both cases studied. Therefore, additional information regarding CTs role was required hence the interviews had to be complemented with the observation and artefact analysis to better understand the role of CTs (e.g. use of spreadsheets). Some strategic drivers to adopt and use collaboration technologies (such as Microsoft partnership in Case 2) also played a role in how requirements collaboration was supported, something that was not foreseen prior to conducting this research.

Furthermore, in the initial stages of the data collection in Case 1 the researcher felt the need to gain a better understanding of the project context and the collaboration challenges faced by the stakeholders beyond the immediate project team. Therefore, more participants (mainly members of the steering committee) on the client side were interviewed. Some of the interview questions were modified based on their role in the project while keeping the focus on project context and change management process. This could be seen as a potential limitation of the research design; however, these interviews enabled the researcher to gain insights into the project from an additional and broader perspective on the phenomenon of interest. While interviewing these participants, questions related to critical incidents, and (some of the) questions regarding collaborative technology in use were omitted.

As presented then, the thesis builds on the review of literature in the topic area of requirements management (RM), situated especially in GSD environments, and reports a detailed multiple case study analysis. The case by case evaluation followed by a cross case analysis that underlies this research is in line with the broadly interpretive perspective noted in Klein and Myers (1999).

As noted in Chapter 3, understanding practitioners' beliefs through qualitative discourse reflects the researcher's belief that social and organisational reality can only be understood by interpreting the meaning that humans attribute to their actions as they make sense of their world and everyday work (Klein & Myers, 1999).The flexibility of a multiple case study design provided the opportunity to gather data from multiple sources and from two different cases, which informed a more holistic understanding of the RCM practices and the role of technology in a GSD context

There is a common recognition that case studies should be evaluated against criteria fitting to their nature (Klein & Myers, 1999; Walsham, 2006). The seven principles for evaluating interpretive research proposed by Klein and Myers (1999)[21] were used to evaluate this research which are embedded in Table 6.3. Because of the interpretive nature of the case study work carried out for this research i.e., presenting interpretation rather than 'fact', producing a cogent and credible account for the reader is essential. Walsham & Sahay (1999) suggest that, to produce convincing explanations of phenomena of interest, interpretive case study accounts need to demonstrate authenticity, plausibility, and criticality. To that end, this case study is evaluated against these three criteria, as summarized in Table 6.3..

Criterion	Explanation	Application in this research
Authenticity	"Ability of the text to show that the authors have 'been there', by conveying the vitality of life in the field" (Walsham, 2006, p. 326)	• Describes the nature and extent of the fieldwork, the researcher's role and their interaction with participants [A multiple case study of two real life software development projects with over 11 months of data collection in two phases from the participants involved in dealing with changes requirements in two GSD projects. Chapter 4 & Chapter 5 Obtaining first-hand information from participants using semi structured interview and Unobtrusive observations]. • Displays familiarity with participants' everyday actions through interviews, project and RCM related artefacts.

[21] (*1. The Fundamental Principle of Hermeneutic Circle, 2. The Principle of Contextualization, 3. The Principle of Interaction Between the Researcher and the Subjects, 4. The Principle of Abstraction and generalization, 5. The Principle of Dialogical Reasoning, 6. The Principle of Multiple Interpretations* and *7. The Principle of Suspicion*) (Klein & Myers, 1999)

		•Uses thick descriptions and utilizing over two hundred direct quotes from the research participants. (Chapter 4 & 5) • Presents perspectives of multiple participants both from vendor and client side and is sensitive to different participant interpretations (Chapter 4 & 5) • Demonstrates systematic, disciplined and iterative approach to data collection and analysis with two phases of data collection, member checking and validation (Chapter 3) • Avoids imposing preconceived notions on the data and allowed data to inform personal perspectives by adopting exploratory rather than explanatory approach to research (Chapter 3)
Plausibility	"How well the text connects to the personal and professional experience of the reader" (Walsham, 2006, p.326)	• Uses schematics such as tables, figures, models and visual mapping to make sense of the data for the reader • Uses a 'theoretically aware' approach, drawing on concepts from activity based computing theory (Chapter 3, Chapter 4 and Chapter 5) • Builds on a comprehensive understanding of contemporary global RCM practices, informed by content and context (Chapter 4, 5 and 6) • Considers the social and historical context of the phenomena investigated (Chapter 4 & 5) • Demonstrates the relevance of the analysis to contemporary RCM practices (Chapter 6) • Makes recommendations for organisations involved and general practitioners in GSD work for evaluation based on their experience (Chapter 7) • Makes distinctive contributions by introducing the concept of informal change distilled from the cross case findings and developing synthesized models for managing informal changes (Chapter 4 & 5) • Contributes to the existing body of knowledge by extending an existing model of GSD challenges impacting RE activities to include RCM activities (Chapter 6)
Criticality	"The way in which the text probes the reader to consider their taken-for granted ideas and beliefs" (Walsham, 2006, p. 326)	• Offers a novel model for understanding informal change management practices as a natural practice in contract based software development (Chapter 6) • Applies non-mainstream ways of thinking about RCM practices in GSD environments and highlights the "taken for granted" aspects of collaborative technologies in managing requirements change especially for RFP based projects. (Chapter 6 & 7)

Table 6.3: Research Evaluation

6.6 Chapter Summary

This chapter has discussed the newly identified challenges in GSD projects based on the findings from two cases studied for this thesis. It presents a model of confirmatory and newly identified GSD challenges and their impact on requirements change management

activities in each of the studied case. The chapter also discusses a synthesized 'Dual Formality' (DFRCM) model for requirements change management which is based on the actual (in practice) change management processes in the two cases studied. The model highlights the reality of informal requirements change, its prevalence in the studied cases and the activities to manage them.

The chapter also discussed the role of collaborative technologies in managing requirements change activities. The perceived help and hindrances of various collaborative technologies are enlisted and discussed. The chapter provides reflections on the perceived adequacy and efficacy of the existing technologies as reported by the participants. It also discusses the role of spreadsheets as a dominant collaborative technology used in requirements change management in both cases studied.

Chapter 7. Conclusions

This chapter concludes the thesis. Section 7.1 briefly revisits the motivation for the research and the research questions posed. Sections 7.2 and Section 7.3 provide reflections on the main findings revealed in this study based on the two research questions. Section 7.4 outlines the novel research contributions made by this work. Then the subsequent Section 7.5 provides consideration of the limitations of the study and threats to its validity, reliability and credibility. A summary of the implications for practice and research are presented in Sections 7.6 and 7.7 respectively that suggest potential avenues of future work. Finally, the summary in section 7.8 concludes the chapter.

7.1 Motivation Revisited

The research reported in this thesis sets out to explore the challenges faced by practitioners in managing the lifecycle activities of requirements change in GSD projects and the associated role of collaborative technologies.

More particularly, this study sought to understand why managing requirements change is difficult in GSD projects, despite ongoing research efforts and the availability of an extensive body of knowledge. Practitioner perceptions regarding the helpful or hindering role of CTs to manage requirements change were analysed to better understand and minimize distance related challenges. Understanding challenges and the role of CTs (based on empirical findings) was aimed at supporting the ongoing efforts to improve the change management process and to re-evaluate the required CT support to manage requirements in GSD.

Another motivation was to develop a model based on the core requirements change management activities to support cross site process alignment and enhance process visibility. Although researchers have been investigating the topic of requirements management in GSD for over a decade, there is a relative paucity of understanding of the lifecycle activities of requirements change and the role of associated CTs in a GSD context, few notable exceptions are (Carlshamre & Regnell, 2000; Heindl, Reinisch, & Biffl, 2007; Hussain, Buchan, & Clear, 2014; Hussain & Clear, 2014). This research has sought to address this void through an exploratory multiple case study of two organizations involved in GSD work. Specifically, the answers to the following questions were pursued in this research:

RQ1. What challenges do practitioners face while managing lifecycle activities of a requirements change in globally distributed development projects?

RQ2. What is the role of collaborative technology in supporting or hindering lifecycle activities of a requirements change in global software development projects?

The research resulted in several findings reported in chapters 4 and 5 that are discussed in chapter 6. The exploratory nature of the investigation in this thesis provides adequate empirical evidence regarding the actual challenges and the role of collaborative technology to aid research efforts in devising solutions to RCM problems in GSD. The following section reflects on the main findings of this thesis based on the insights gained through the multiple case study and offers some recommendations for research and practice.

7.2 Main Findings Related to Research Question 1

7.2.1 Existing Challenges Verified in the Case Study

In an effort to understand the perceived GSD challenges in managing RCM activities seven practitioners from the two studied cases were interviewed. The focus of inquiry was related to understanding the project context, prescribed versus practiced requirements change management processes or practices as well as the challenges faced by the participants due to distance (geographical, temporal and socio cultural distance).

Participants of this research reported several challenges that were directly or indirectly related to RCM activities that were amplified due to the globally distributed nature of their projects. The outcome was primarily confirmatory – most of the identified challenges had been noted in prior studies and were verified to be present in the two cases studied. This study thus confirmed thirty-two challenges reported in the GSD literature as having a direct or indirect impact on RCM activities. These challenges were placed under six known (high-level) GSD challenge categories (namely *Geographical Distance, Temporal Distance, Inadequate Knowledge Management, Inadequate Communication and Socio Cultural Difference*). These categories of high level challenges were interlinked with each other forming a network of influence on other challenges. Table 7.1 summarizes these challenges already reported in GSD literature but experienced in this research.

Challenge Category	Challenges Confirmed in this Research
Geographical Distance	▪ Communication Overhead (Mockus & Herbsleb, 2001; Jim et al., 2009) ▪ Manage Project Artefacts (Changes, Awareness) (Jim et al., 2009) ▪ Manage Dependencies (Teams & Tasks) (Mockus & Herbsleb, 2001) ▪ Proper Scope Definition (Nurdiani, Jabangwe, Smite, & Damian, 2011) ▪ Ensure appropriate tool support and common set of tool (Nurdiani, Jabangwe, Smite, & Damian, 2011; Prikladnicki, Audy, Damian, & de Oliveira, 2007; Sinha, Sengupta, & Chandra, 2006) ▪ Manage Distributed Stakeholders commitment, interest and relationship ▪ Cost (Travel & Logistics) (Cheng & Atlee, 2007)
Temporal Distance	▪ Effective Meetings (Mockus & Herbsleb, 2001; Cheng & Atlee, 2007; Nuridani et al., 2011) ▪ Reduced Communication frequency and quality (formal and informal) (Mockus & Herbsleb, 2001; Cheng & Atlee, 2007; Ebert, 2011) ▪ Delay (Herbsleb & Mockus, 2003) ▪ Managing Requirements in distributed settings (Damian & Moitra, 2006)
Inadequate KM	▪ Common Understanding of Requirements (Cheng & Atlee, 2007; Mockus & Herbsleb, 2001) ▪ Lack of Team Cohesion (Nurdiani et al., 2011) ▪ Manage Conflict and have open ended discussion of interest (Nurdiani et al., 2011) ▪ Awareness of changes in project artefacts (Damian et al., 2003; Herbsleb & Mockus, 2003; Lanubile, Calefato, & Ebert, 2013)
Inadequate Communication	▪ Appropriate participation from system users and field personnel (Bhat, Gupta, & Murthy, 2006; Bhat et al., 2006) ▪ Trusting Working Relationship (Mockus & Herbsleb, 2001) ▪ Awareness of Local Work Context (Damian, Chisan, Allen, & Corrie, 2003; Omoronyia, Ferguson, Roper, & Wood, 2010) ▪ Poorly Defined Requirements (Cheng & Atlee, 2007) ▪ Faulty Effort Estimates (Nurdiani et al., 2011) ▪ Limited face to face meetings (Mockus & Herbsleb, 2001)
Socio-Cultural Difference	▪ Diversity in Customer Culture and Business (Nurdiani et al., 2011) ▪ Terminology Difference (Prikladnicki et al., 2007) ▪ Lack of Shared Understanding (Mockus & Herbsleb, 2001) ▪ Language Difference (Cheng & Atlee, 2007) ▪ Cultural Difference (Cheng & Atlee, 2007)
Organizational Diversity	▪ Process Misalignment [Define, Implement & manage joint processes] (Prikladnicki et al., 2007; Ebert, 2011) ▪ Integration of Unequal Domain Knowledge (Schmid, 2014; Herbsleb, 2007) ▪ Inadequate Skill Set (Schmid, 2014) ▪ Inconsistency in work Practices (Cheng & Atlee, 2007; Prikladnicki et al., 2007; Mockus & Herbsleb, 2001) ▪ Limited Experience (GSD & Working together) (Herbsleb 2007) ▪ Diversity in Process Maturity (Ebert, 2011)

Table 7. 1 Known GSD Challenge Categories Verified in this Case Study

However, there were several challenges that were either not experienced, reported or had no impact on RE and RCM activities in the two cases. These challenges included: *data loss during transfer, security breach*, *fear of job loss (Nurdiani et al., 2011)*, *customer employee unwillingness to collaborate relatedness with other suppliers*, *(Šmite, 2006)*, *(Nurdiani et al., 2011)*, *and differences in Laws and Regulations in different countries (Schmid, 2014)*.

7.2.2 New Challenges Identified

Analysis of the collected data for this research revealed some new challenges that impacted RE and RCM activities. These issues were in addition to ones reported in Table 7.1 and exacerbated the existing challenges especially because of the globally distributed nature of the projects. The challenges include *1) Immaturity of the product 2) Lack of Product Knowledge 3) Client Led RE 4) Inadequate Vendor Expertise Onshore 5) Powerful Proxy 6) Heavy Documentation 7) High Level Customization Requirements 8) InfRC & Inappropriate RCM Process*. The impact of these newly identified challenges varied across the two studies. Table 7.2 lists down these challenges including the severity of their impact on each of the studied case.

The impact of a challenge was considered high if it affected more than five requirements and change related lifecycle activities. The impact was seen as medium if the challenges affected three to five activities and was taken as low if they affected less than three activities. For example, Case 2 had a relatively more mature product and the development team involved had better knowledge about the product as compared to Case 1; in Case 1 there was no proxy client so no challenges related to the proxy client were experienced; Case 2 did not have the problem of insufficient vendor expertise onshore with the client; and only in Case 1 was the problem with the client-led RE process experienced.

New Challenges Identified	Case 1 Impact	Case 2 Impact
1. Immaturity of the product	High	Low
2. Lack of Product Knowledge	High	Low
3. Client Led RE Process	N/A	High
4. Inadequate Vendor Expertise onshore	High	N/A
5. The Powerful Proxy Client	N/A	High
6. Heavy Documentation	High	Med
7. High Level of Customization Requirements	Low	High
8. InfRC & Inadequate RCM Process	High	High

Table 7.2: New Challenges Identified in this Case Study *N/A = Not applicable*

The findings from this research thus highlight some new challenges in managing requirements changes. These new challenges in relation to RQ1 are now briefly described.

7.2.3 High Impact Challenges in Case 1

7.2.3.1 Immaturity of the Product

Product immaturity[22] impacted requirements elicitation, analysis and specification in Case 1. The challenges arising due to product immaturity included difficulties in product evaluation, feature identification, change definition and change implementation. For the client, inadequate product documentation and limited user base meant that extra effort was required in product evaluation which contributed to an inaccurate mapping of user requirements to existing product features. In addition, feature incompleteness in the selected product (Case 1) resulted in significant misalignments (or gaps) between existing product features and client organization's processes as well as user requirements. This gap had to be bridged by software intensive customizations which are considered a challenge even for a collocated team but proved to be a bigger challenge for distributed teams.

7.2.3.2 Lack of Product Knowledge

Lack of product knowledge (in combination with limited domain knowledge) had a significant impact on RCM activities especially in the GSD environment as experienced in Case 1. It hindered the client's ability to understand the vanilla product, perform adequate reviews; carry out effective effort and cost negotiations with the vendor and accurately identify gaps between product feature offerings and business requirements. Limited product knowledge also became a significant challenge while testing the updated versions of the product because the client testing team found it hard to identify the changes and compare them with the existing product functionality.

Furthermore, the limited understanding of the product and its existing features did not allow the client BAs to comprehend and appreciate the implications of the proposed product customizations. The knowledge of these implications was crucial for effective planning, prioritization and negotiations with the vendor.

[22] Determined on the basis of: product (age), feature in/completeness, supporting documentation and existing users

7.2.3.3 Client-Led RE Process

Client-led RE activities created a void between the vendor team and the client project team members that resulted in challenges for RCM activities. The incompatibility of domain terminology and unequal product knowledge (between the client and the vendor BAs) created a challenge in developing a shared understanding of the requirements. This was considered as the client's responsibility due to their lead role in RE. Lack of both domain and product knowledge on the client side due to use of generic BAs hired for the project was an impediment to translating business and user needs. It also hampered the client's ability to convert user needs adequately into an agreeable representational format for the vendor development team. This caused a more pronounced need for collaboration for requirements review, analysis and negotiations with the vendor BA that resulted in additional coordination overhead.

Due to the limited opportunities for synchronous communication both client and vendor team members relied heavily on the asynchronous means but given the poor client-vendor relationship asynchronous means were inadequate for conflict resolution. Personality clashes and professional rivalry arose during requirements or change analysis, review and negotiations due to similar roles on both the client and vendor site (project owner and product owner, business analysts) as cross-site team members were stepping on each other's toes.

7.2.3.4 Inadequate Vendor Expertise Onshore

The unavailability of onshore vendor representation and expertise in the form of a 'boundary spanner' or 'intermediary' role led to several requirements related challenges. It did not allow the client to engage in face-to-face communications with a competent conversation partner from the vendor side to discuss critical issues especially related to requirements. Transfer of requirements related knowledge was significantly hindered and the communication gap was widened because of the absence of any onshore vendor personnel. The client team members especially in Case 1 faced challenges during requirements negotiations as they could not satisfy product related queries and clarify their assumptions with a vendor representative having the desired product knowledge. The value of onshore vendor representation such as facilitating knowledge transfer, coordinate more efficiently and contributing to project success are well known and hence 'Zero' onshore presence (as found in Case 1) is exceedingly rare in GSD (Srikanth & Puranam, 2014) as observed in Case1. However, in Case 2 a powerful vendor

representative was helpful in requirements related knowledge transfer, facilitating meetings and bridging communication gaps. The challenges due to the excessive powers bestowed on the proxy client carrying out this role created its own challenges discussed in more detail in the next Section 7.2.4.

7.2.4 High Impact Challenges in Case 2

7.2.4.1 The Powerful Proxy Client

The role of the powerful proxy client, and the negative impact it had on requirements and change management activities, was a key finding from Case 2. The proxy client, also the CEO of the vendor company, was a *'tech-savvy'* domain expert with excellent communication skills and socio-cultural understanding of the business norms in the client country. The aforementioned qualities of the proxy client and his influential position provided greater authority and autonomy to his role and created a power imbalance between him and the other team members. His role (comparable to a boundary spanner or an intermediary (Boden & Avram, 2009)) generally had a positive influence on the requirements-related activities. The proxy client facilitated cross-site communications, awareness and knowledge transfer in distributed collaboration.

Since the development team was dependent on the proxy client both for resources and coping with important uncertainties (two main forms of power (Sauer, 1993)) this resulted in power concentration in the proxy client's role. However, the concentration of additional authority in the proxy client's role had a negative impact on requirements and change management activities. The issues caused by the polarization of power in favour of the proxy client included dealing with a constant flow of informal change requests by the proxy client, the need for ongoing negotiation of effort estimates with him, and dealing with misrepresentation of the (actual) client's requirements caused by the proxy client. This key finding highlights a gap in the existing literature which sees the role of boundary spanners or intermediaries only in a positive light (Nguyen-Duc et al., 2014; Nisar & Hameed 2004).

7.2.4.2 Heavy Documentation

Explicit documented knowledge demanded effort investment from team members especially when working in a GSD environment. Dealing with government clients and conforming to certification guidelines (e.g. CMMI), there was a pronounced need to document processes and activities and practices in Case 2; as the price of working in GSD

environments (Fowler, 2006). However, continuously changing requirements that required heavy documentation effort and overhead compelled people to use short cuts and skip 'unimportant' documents. Identifying the right balance between important and unnecessary documentation was not trivial and led to changes in documentation practice whereby important documents were skipped resulting in problems for downstream implementation and testing activities related to requirements change (Sabaliauskaite et al., 2010).

7.2.5 Common High Impact Challenges (Case 1 and Case 2)

7.2.5.1 High Level of Customization Requirements

High level of customization requirements implicitly noted in the contract (Case 1) caused significant challenges when these requirements were later elaborated and verified. The teams found it difficult to separate customization or changes in requirements from the agreed contractual requirements due to the high level of requirements. Furthermore, requirements elaborations and the resulting scope negotiations demanded frequent prioritization and negotiation meetings. Coordinating such meetings and carrying out requirements related collaborations over distance in the absence of a good client-vendor relationship was challenging especially in Case 1. The infrequent, untimely and ineffective meetings over these customizations resulted in several problems not only for requirements related activities but also for change implementation and verification activities.

7.2.5.2 Inappropriate RCM Change Management Process and InfRC

Issues caused by following an inadequate change management process were harder to address in a GSD environment. Inadequate planning and anticipation of requirements change frequency led to the application of unsuitable change management practices which were not aligned with the development methodologies chosen in both studied cases.

For example, in Case 1, applying formal change management only for very high level changes and leaving low impact change management unduly flexible resulted in overwhelming number of changes and a high cost for the project. Instead of following some defined practices in Case 1, prioritization and scope negotiations were used as the main mechanisms to handle both formal and informal requirements changes. In many cases it caused frustrations for one or the other party involved. Similarly, a very weak

internal change management process that allowed any number of informal and internal changes to be included and implemented within the same time and scope created many (cost, budget and schedule related) challenges for the project and also for managing RCM activities in Case 2.

The identification of an *informal requirements change* process as a phenomenon actively carried out in practice, parallel to any formal change management process, is a novel finding from the two cases studied. Informal requirement changes and their management can be contrasted with the classical perspective of requirements change management which involves highly formalized and rigorous change processes (Sommerville, 2011).

Informal requirements change represents an under-reported and variably managed project dimension of hidden work which demands suitable buffers to accommodate such activities in development and change management processes. The informal requirements change management processes found to be evident in both studied cases directly impacted RE and RCM activities. It resulted in overburdening the development team that impacted project schedule and caused frustration among team members. It also required additional communication overhead to resolve issues arising from informal change requests in the studied cases.

7.3 Main Findings Related to Research Question 2- Role of CTs

7.3.1 Helpful Role of Existing Collaborative Technologies and their Efficacy

Collaborative technologies (CTs) were perceived as adequate and mainly useful by the participants for their GSD projects. Distributed collaboration, requirements change management was considered adequately supported by the in-place CTs except for the occasional technical problems which were perceived as 'routine' especially in Case 2. These are contradictory findings to the assertions made in some GSD studies suggesting existing GSE tools cannot support distributed collaboration (Tell & Babar 2011). Similarly, the research participants preferred standalone, task specific/dedicated tools as noted by (Mishra & Alok, 2011; Salger et al., 2010) rather than having a suite of collaborative tools and technologies under a uniform platform as suggested by (Tell and Babar (2012). It cannot however be concluded that these findings would be applicable to any project irrespective of its size and the suite of existing collaborative technologies used. For projects with significantly larger number of lines of code and greater collaboration needs the above mentioned set of CTs might not suffice.

7.3.2 Hindering Role of Collaborative Technologies

Collaborative technologies were also found to create some hindrance for the practitioners while they carried out activities related to RCM. The notable hindrances especially for non-technical users were: dealing with the complexities of the collaboration tools in terms of their setup structure, complex user interface, information storage and retrieval due to complexity of interface and huge volume of information stored in the knowledge repositories (*Confluence* and *JIRA*).

Communication and collaboration technologies at times hindered collaboration due to poor audio quality, latency and echo resulting from inadequate connection speed. These problems often caused message misinterpretation, delay, confusion and frustration. Similarly email messages were sometimes misinterpreted causing problems between the sender and the receivers. Furthermore, emails were found to be an unsuitable communication media for negotiations and conflict resolution. In conflicts, emails actually contributed more to the complexity of the situation and challenges rather than resolving them. The participants preferred to use synchronous media for negotiations which 'rich' communication technology with a high *social presence*, *symbol variety* (ways in which information may be communicated), as noted by Massey, Hung, Montoya-Weiss, and Ramesh (2001).

Overcoming technology preferences (spreadsheets over JIRA) was also found challenging for managing requirements especially in Case 1. It caused the client to invest additional time and effort which resulted in frustration and poor client-vendor relationship. Preference of spreadsheets over *JIRA* and *Confluence* contributed to wasted effort in recording different versions of documentation on multiple media that generated isolated and contradictory data pools. Maintaining parallel documentation on these media required additional effort and also caused information fragmentation. Recording information across several media further resulted in inconsistent views and lack of shared understanding of the requirements.

7.3.3 Role of Spreadsheets as Collaborative Technologies

In the studied cases the role of spreadsheets as a collaborative technology was critical in enabling the collaboration demanded for requirements change management in GSD. Spreadsheets played a vital role in bridging requirements change management activities between distributed stakeholders and the core development team. They became part of the repertoire of both synchronous and asynchronous collaborative technologies for complementary purposes. The gamut of technologies thus formed was inclusive of role

specific and formal (control and bug tracking system) and ad hoc technologies (email and instant messaging) to complement IDEs for the required collaboration in GSD (Cheng, Hupfer, Ross, & Patterson, 2003)

The use of a combination of collaborative tools was contextual in nature and was often influenced by factors such as tool availability, successful experience with a similar technology combination, ease of use and learnability. Spreadsheets as CTs were found useful by the practitioners because their contents easily fitted within a broader repertoire of collaborative technologies (Hussain et al., 2014). Spreadsheets also played a critical and supportive role as repositories for storing requirements related artefacts and afforded the ability to share these embedded requirements artefacts across sites for both synchronous and asynchronous collaboration. The flexible nature of spreadsheets (allowing modifications to both their structure and content) allowed the accommodation and evolution of cross site requirements change management practices in the studied cases (Hussain et al., 2014). The use and heavy reliance of spreadsheets in requirements and change related activities and collaboration was also possible in both projects studied because of the small to medium nature of projects and relatively small number of software requirements as mentioned previously in Section 7.6.1.

7.4 Research Contributions

This research makes several novel contributions to our understanding of RE in GSD, across the substantive and conceptual domains of McGrath and Brinberg (1983).

7.4.1 Contributions in the Substantive Domain

1) This study has profiled the first known, in depth, multi-case case study exploring the lifecycle activities of a requirements change for two product customization projects in professional environments and GSD settings. The study has resulted in identification of eight new challenges related to requirements change management carried out in a globally distributed development environment.

2) It has added to the very few field studies in the area of requirements change management in global software development. (Hussain & Clear, 2012; Niazi et al., 2008; Sinha, Sengupta, & Chandra, 2006)

3) The study has addressed a gap in the literature by exploring collaborative technology support for lifecycle activities of requirements change management for product customization projects in a global software development (GSD) environment. As opposed to the proposed collaborative technology push in (GSD) project, the findings of this study

have confirmed the adequacy of available collaborative technologies (adapted to local needs in workable combinations) and use of dedicated tools instead of a more generalized suite of collaborative technology.

4) The study has identified the extended role of spreadsheets as CTs in both studied cases and their support of RCM activities.

7.4.2 Contributions in the Conceptual Domain

5) This study has developed a novel theorisation of the notion of "informal requirements change" verified by both cases studied and validated by the practitioners involved.

6) The study has presented an illumination of a requirements change management (RCM) lifecycle synthesized from two models of RCM in practice from the two studied cases in globally distributed development contexts (See Section 6.4 Figure 6.3). This Dual Formality RCM model has the potential to be applicable in contexts similar to those described in this research.

7) The study has produced the first known model that captures in one snapshot various GSD sites, Roles, activities and related artefacts by adapting the Activity Based Model proposed by Korpela et al. (2002) (Chapter 4, Figure 4. 2), and the notation taxonomy for Global Requirements Engineering (GRE) proposed by Laurent et al. (2010) (Chapter 4, Figure 4.1).

8) The study has produced a model (cf. figure 6.1) to represent GSD challenges and their impact on RCM and RE activities by adapting and extending the model proposed by Damian and Zowghi (2003).

7.5 Research Limitations

As is the case in any time-boxed research endeavour this study has its limitations. In particular, access to case study sites and participants was limited due to a range of factors – discussion of these constraints and their effect now follows.

The information in Case 1 was collected predominantly from the perspective of the client organization, with only one interview being conducted from the vendor organization in that case. Therefore, the perceptions regarding the RCM challenges and the role of CT in Case 1 are primarily indicative of the client's perspective. In contrast, the researcher had limited access to the end client in Case 2, which meant that the findings were derived mostly from the vendor's perspective. The client perspective could only be seen through the vendor role acting as the proxy client. In both cases greater exposure to the second

perspective could have provided additional insights. However, it is hoped that sufficient coverage of both perspectives was achieved across the two cases.

In Case 1, the researcher did not know the vendor organization personally and so was introduced to them by the client organization. As such there was limited rapport between researcher and organisation, and this may have negatively impacted on their engagement. Although all six employees in the vendor organization met the requirements of potential participants laid out in the interview protocol, only two of them actually agreed to participate in the research. Furthermore, one of the two (the chief architect) pulled out of this research prior to being interviewed due to lack of time. This was a significant setback, meaning that the views obtained from the vendor organization in Case 1, especially regarding the internal processes, team collaboration, technologies and challenges, are sourced from a single participant which introduces a level of bias in the data. Furthermore, since the participant interviewed was a high level executive, business analyst and principal of the vendor company, the views might not be consistent with the insights that could be gained from development staff in terms of their particular reflections on RCM challenges or the role of collaborative technologies.

In Case 1, some of the interview participants were stakeholders not directly involved in the project development activities, such as members from the steering committee, e.g., internal auditor, research director and IT department head. In addressing these participants, the interview questions were necessarily slightly modified, i.e., asking about general challenges with the project or existing collaborative technology (CT) use, rather than about challenges with the change management process or remote collaboration with the vendor using CTs (since they were not directly involved in it). These interviews helped in gaining deeper understanding of the project context and provided insights into the challenges faced by general stakeholders due to geographical distance between the client and the vendor teams.

In Case 2, all software development activities were performed at the Pakistan site by the (collocated) development team who largely had requirements development and project management related dependencies with the USA site. Given this particular arrangement the challenges experienced by the development team in this case may not reflect other geographically distributed development teams' perspective of challenges faced. In addition, and as noted above, data collection in this case (Case 2) was predominantly done from the vendor organization, with only one interview being conducted with the proxy client; hence the reported challenges will be generally indicative of the vendor's

perspective. However, it is contended that the fragmented organizational set-up of the vendor company, where the US office was officially treated as a client for the offshore development site in Pakistan, allowed some consideration of the client's perspective in the study.

One interviewee did not permit their interview to be recorded, and therefore the researcher had to rely only on the notes taken during the interview which could be seen as a potential limitation for this research.

In some of the Case 1 interviews a two-member interviewing team was used. In these cases, the researcher focused on the interviewee and the official scribe took notes and occasionally prompted where necessary. This ideal situation (Darke et al., 1998) could only be used in a few interviews, however, and at only one interview site, i.e., New Zealand, due to limitations on time and availability of the scribe. However, since all but one of the interviews across both cases were audio recorded, transcribed and later verified by the participants, the possibility of errors and omissions was minimized. It should also be acknowledged that seven out of the 33 interviews were conducted in Urdu (Pakistan's national language) and were then translated into English (Appendix 6). This could have introduced human error in the translation of concepts or perceptions held by the participants. As noted above, however, the translated interview scripts were shared with the participants for correction and the translation of selected material was also verified by the researcher's colleague who was a native Urdu speaker, to avoid gross errors.

This study was conducted in two GSD settings, permitting a degree of abstraction and synthesis into a single RCM process model (Chapter 6, Section 6.4, Figure 6.3). However, it cannot be claimed that the derived and synthesized model is 'complete', or reflects RCM in other contexts. In fact, it does not, yet, include the global aspects (sites and roles) as identified in a previously proposed requirements change management model by the researcher (Hussain & Clear, 2012). Although the Dual Formality RCM (DFRCM) model (Chapter 6, Section 6.4, Figure 6.3) itself has not been validated, however the individual RCM models used to produce the DFRCM model were validated by the practitioners from each participating organization. DFRCM in Chapter 6, Section 6.4, Figure 6.3 is the final version updated on the basis of participant feedback. Validation for the model in Chapter 6, Section 6.4, Figure 6.3 however, fits the type "evaluation" according to Shaw, (2003) for an empirical model because it "generates results that fit actual data" in addition to a) adequately describing and b) accounting for the phenomenon of interest- the other two criteria for validation of models.

As for the research, evidence from multiple cases is generally considered more compelling and the findings more robust. In this research it allowed cross-site comparison without necessarily forfeiting within-site understanding (Herriott & Firestone, 1983). Therefore, multi-case research design broadened the scope of this research and avoided findings that otherwise could be "wholly idiosyncratic". Using empirical data from multiple cases to develop the synthesized DFRCM model allows for model generality as the results build on and confirm evidence gathered from one case to the other. This approach is described by Kock et al., (1997) who suggest that number of cycles (or iterations) are directly proportionate to research scope and generality of the produced models.

The DFRCM model (Chapter 6 Section 6.4, Figure 6.3) is contextualized to reflect the realities of RFP-based product-related requirements change management process activities. It still needs additional work to incorporate the global dimensions such as technology, artefacts, multiple roles and communication across geographically distributed sites. Both the notion of informal change and its capturing DFRCM model developed through this exploratory study are insights as hypotheses for future explanatory studies to be carried out in this direction. Validation of this work might take the form of a survey of expert practitioners in the field of RCM in GSD.

The findings reported are derived mainly from data collection through interviews and the review of electronic and non-electronic artefacts, with very limited data collected through observation. Although additional observation data from the field might have strengthened the findings, given the inherently constrained scope and magnitude of the research, spending further time in observation in the two case study sites was not feasible. Subsequent observational studies as well as other forms of case study are needed in order to assess the more general validity of the findings and their wider relevance and applicability in other settings.

7.6 Implications for Practice and Recommendations

There is no escaping the fact that global software development is difficult (Ebert, 2014). It has again been made evident from the findings of this study in terms of the 32 confirmatory GSD challenges faced by the practitioners and their perceived impact on change management activities (see Section 7.2, Table 7.1). GSD brings with it certain essential issues due to time zone, language, cultural and organizational differences as well geographical distance, issues found to be relevant in the two cases studied. The clients

and vendors involved in GSD contracts need to be flexible to adapt their processes and to be prepared for the additional challenges before embarking on the journey of GSD.

Based on the insights gained from this study regarding the challenges that can arise in managing requirements change the following recommendations are provided to practitioners. In contexts similar to those considered in the two case studies these recommendations could address some of the issues around RE that impact RCM activities in GSD-based product-customization projects. While it is fair to say that many of these recommendations are 'known' – that is, they make intuitive sense and have been recommended elsewhere – their specific relevance to product customization in the GSD context, coupled with their solid grounding in empirical evidence, make them worthy of restating here.

7.6.1 COTS Selection and Customization

i. Adhere to at least one available product selection technique while selecting the product

Substantial effort should be invested in product (and vendor) selection since the misalignments between the product offerings and the client organizational needs are likely to result in both collaboration and development intensive effort. Such collaborations are additionally risky in a distributed environment. There is a vast body of knowledge around pre-packaged software that could aid in product selection but the key is to select (based on the project context) and adhere to one of the available product selection techniques (Ibrahim, Elamy, Far, & Eberlein, 2011; Mohamed, Ruhe, & Eberlein, 2007a; Ruhe, 2003).

ii. Avoid the customization trap

Heavy code customization in a GSD environment is quite challenging due to distance, constrained communication channels and informal interactions between the client and the vendor (Hussain et al., 2014). Careful product evaluation and selection should be used as a measure to avoid huge mismatches between the product features and business case and project goals.

A customization intensive COTS project can morph the initial scope and plan into a complex, expensive and risky endeavour especially in distributed development settings (ICAC, 2013). Keeping a tight link between the initial business case and the deliverable as well as having a clear idea of up-front scope can help avoid undue customization.

Many COTS customization troubles are unavoidable as they arise only after an imperfect ''replication'' is implemented. This is due to the inability of the stakeholders to formalize

their practices completely in a timely manner. However, the additional instances of ‘‘bad fit’’ emerging from deeply sedimented implicit users’ knowledge should not cause the client to become too ambitious in product customization (Rowland & Gieryn, 2008).

iii. Use existing product (COTS) documentation to draft baseline for customizations

Availability of adequate existing product documentation is highly desirable for customizations needed in a COTS product as it provides a baseline upon which the desired changes can be built. Having adequate documentation should reduce the client’s reliance on the vendor’s product knowledge for identifying the gap (misalignments) between user requirements and the existing features. It saves the client’s additional effort on evaluation, elicitation and later negotiations with the vendor. In addition, a documented baseline of the existing features aids in identification and testing of the newly implemented customizations. Testing a COTS package without a baseline to test the changes against is very difficult.

Even in the case of mature products, baselining features through existing documentation can help in reducing unanticipated communication and coordination required for distributed negotiations prioritization of requirements and scope adjustments that may result in delays, bad client-vendor relationship and other downstream activities. Such challenges get exacerbated in globally distributed projects due to collaboration barriers introduced by distance, cultural diversity, time differences and inadequate communication.

Being too ambitious with the scope will result in over-stretching the capacities of the client and vendor teams. Furthermore, adapting some of the client business processes to the product offerings can help bridge the mismatches and avoid a customization trap. (ICAC, 2013).

iv. Critically examine product maturity before purchasing

Purchasing a product with incomplete or unstable features, little supporting documentation and a limited existing user base can lead to extensive customization, bringing an additional challenge to the existing complexities in GSD projects (Noll et al., 2010). Too many customizations to align the product to a particular context increases the risk of project cost and schedule overruns and decreases its reliability (Torchiano, Jaccheri, Sørensen, & Wang, 2002). At the same time, negotiating and managing changes in a globally distributed environment with constraints on overlapping time, travel budgets, and face to face communication is extremely challenging.

7.6.2 Project Management

v. Align project management practices across the vendor-client boundary.

In distributed contexts the RE activities are driven largely by the client and are a substantial part of the project effort. This means the rhythm of work from the client team is closely interlinked with the vendor's development efforts. A misalignment of project management practices will interfere with the coordination of activities, as observed in case 1 of this study.

vi. Increase availability of product expertise for the client

Unavailability of the vendor's product expertise can be a bottleneck, delaying negotiations and decisions especially if the vendor is a start-up company. Ensuring that more than one product expert is available on demand should obviate this issue. Should this not be possible due to cost constraints, there need to be regular vendor representative visits to the client site to somewhat reduce the lack of onsite product expertise. As seen in Case 1, a single visit was not enough.

vii. Have a vendor representative onshore with the client organization

At least one vendor representative should be made available onshore and ideally onsite with the client organization, to facilitate requirements related communication, knowledge transfer and also to avoid unnecessary delays and communication gaps. If such an arrangement is not financially viable, project planning for projects of this scale should include at least two onsite visits by a vendor representative.

viii. Avoid polarization of power in favour of boundary spanners or intermediaries

Authority and autonomy in the boundary spanner or intermediary roles is necessary to have credible vendor representation and sharing knowledge between the client and offshore development team (Boden & Avram, 2009). However, organizations involved in GSD work should acknowledge the inherent risks of such a role design as too much concentration of power in these bridging roles can have a negative impact on the project. Highly skewed and undue empowerment of such roles should be avoided so that exploitation of roles with relatively less power is avoided or at least reduced.

From an organizational perspective this situation however can also be viewed in a positive light and providing a potential opportunity for their growth. When relatively successful organizations have their owners in highly empowered bridging roles, they sit on the verge of the next era of organizational growth called delegation (Greiner, 1989). If a successful decentralized structure is applied this organizational growth area proves useful for obtaining organizational expansion through heightened motivation at lower levels. This growth however, can only result through a decentralized organizational structure which

delegates authority to managers and offers greater responsibility and incentives to achieve organizational goals (Greiner, 1989).

ix. Establish and retain a trustworthy relationship with the clients

Vendors need to maintain their credibility with the client regarding their claims, product features and delivery promises in general. Trust and a good relationship was reported in this study as a necessary condition for conflict resolution and was considered vital in establishing strategic alliance (Gallivan & Oh, 1999). Due to their superior product knowledge the vendors enjoy a dominating position in product development or customization related negotiations with the client, however they can lose their credibility and trust if the product feature claims are not verifiable when the products are evaluated. Therefore, vendors need to establish and retain a trustworthy relationship to avoid undermining the collaborative relationship and ultimately the success of the project (Hussain et al., 2014).

7.6.3 Change Management

x. Anticipate change frequency and carefully design a requirements change management process in line with the development methodology

Carefully plan for and anticipate the frequency of requirements change so that a suitable change management process can be adopted that is in line with the development methodology. A formal agreement to establish and follow a common change management process should help improve task coordination and visibility among vendor and client stakeholders and across site boundaries.

xi. Acknowledge and effectively handle informal change management activities

Acknowledging informal change-related activities being carried out in parallel with the formal change management processes is essential to the handling of changes. Allowing a phase for requirements elaboration and clarification, even after the formal contract sign off, and subsequent modification and agreement on the project scope may ease challenges caused by rigid and formal change management processes.

In order to avoid overburdening the development team with a large number of informal requests (albeit small in terms of effort required for each request), informal change requests should be recorded and budgeted to a threshold. Once the threshold is reached the client should be informed to compensate for any additional work required to implement the informal change requests. The application of practices from agile methodologies, such as Scrum or Disciplined Agile Delivery (DAD) (Ambler, 2014), that embrace change and are more accustomed to handling change can be applied to manage informal change more effectively. A streamlined but flexible approach that avoids

unnecessary early investment in detailed documentation (including traceability) can help reduce the negative impacts of informal change requests. Scrum and DAD apply prioritization as a practice and a mechanism to deal with requirements changes effectively and develop high value software (Ambler, 2014).

xii. Encourage vendor led requirements engineering activities for Packaged Software

Following a more conventional strategy in the development or customization of packaged software, a vendor–led or market-driven requirements engineering (RE) approach should be encouraged (Jarke et al., 2010). Client-led RE activities in case 1 (NZ) created a void between the vendor team and the client stakeholders. It can also lead to professional rivalry and personality clashes due to organizational and cultural differences.

7.6.4 Collaborative Technologies

Based on the findings from the two studied cases in this research; the following recommendations for practice should be considered when planning a repertoire of CT support in GSD.

xiii. Be open to the notion of collaboration technology as an enabler rather than a preventer of GSD or collaboration challenges.

It is difficult to define the notion of a 'team' in GSD environments and thus to decide on the generic collaboration support that suits every project context. Geographically distributed teams do require adequate tool support for their work collaboration, however, an undue focus on collaborative technology that can solve 'all' collaboration related issues, should be avoided. Collaborative technology thus should be seen as an enabler and supporter of collaboration practices rather than a preventer of challenges.

xiv. Avoid a single 'heavyweight' collaborative technology for all stakeholder collaboration needs

In contrast to the recommendations by Tell and Babar (2012), to use a collaboration technology on a single platform, almost all cross-site team members (in both cases) were satisfied with the existing repertoire of the collaborative tools. Furthermore, they preferred using a combination of dedicated as well as unspecialized tools to perform their collaboration tasks.

In fact, the use of a heavyweight collaborative technology (such as Team Foundation Server and a combination of JIRA & Confluence), made the learning as well as the usage of the technology more difficult especially for the non-technical stakeholders. A single platform collaboration tool with an even more comprehensive set of features as suggested by Tell and Babar (2012), would be more difficult to learn and use without involving a

steep learning curve. Such technological support for collaboration should therefore should be very carefully planned based on the project's need taking into consideration existing platform experience, standards and practices or should be avoided altogether.

xv. Administer adequate employee training before introducing non-trivial collaborative technologies into projects

Introducing new collaborative platforms for a new project demands adequate training of project team members. The use of collaboration tools such as Microsoft Team Foundation Server or (a combination of) JIRA and Confluence in a project without adequate training leads to underachieving on CT's potential and makes collaboration on project tasks even more difficult. Effective and productive use of CTs can be supported through activities such as *establishment* (setting up and continuously supporting technology use) as well as *reinforcement* (training, monitoring and following up to reinforce established guidelines for use) and *adjustment* (adjusting definitions and guidelines of usage based on member feedback) (Orlikowski, 1996). As observed in the studied cases, team members either use minimal and easy to use feature sets or use features incorrectly due to lack of knowledge and training (for example in code versioning) leading to inefficiency and confusion in collaboration.

xvi. Look beyond the core development team to the CT support needs of cross functional stakeholder groups

Managers should carefully design collaborative technology support to consider all types of stakeholders. This may involve consciously designing the most effective role for collaborative technologies, for example team collaboration software, integrated development environments, emails and spreadsheets, as bridging mechanisms between technical and non-technical stakeholder groups in the wider CT repertoire.

xvii. Pay conscious attention on the activities of technology use mediation (TUM) such as establishment and reinforcement of patterns of use for all collaboration technologies (CT repertoire) used in the project.

Team collaboration technologies introduction in organizations need more conscious attention as they often entail a steeper learner curve and the possibilities of misuse (Kowark & Plattner, 2012). Carefully applying TUM's activities (Orlikowski, 1996), such as *establishment* of guidelines and roles, determining and building consensus around the use of communication and collaboration technologies can help in CT use by the members. Similarly, *reinforcement* activities such as training, monitoring and following up with the users to reinforce the established guidelines of use may minimize issues of use and enhance productivity.

While it is more demanding to set up a suite of integrated tools, the lesser demand for set up and tailoring of collaboration technologies such as spreadsheets still exists. Spreadsheets as part of the repertoire of collaboration technologies can more readily evolve and offer the ability to 'fail fast' because they can be redesigned or tweaked easily to support evolving practices.

7.7 Implications for Research

The synthesized model for RCM presented in this study, which highlights the prevalence and importance of informal change management practices, needs further investigation and validation. This presents a natural extension of this work that could be taken up by future researchers.

The two individual change management models (Chapter 4 Figure 4.9 and Chapter 5, Figure 5.7) are validated by the participating organizations in each case for their effectiveness to map capturing both the formal and informal requirements change management practices. Some changes were also made to the model in Case 2 based on the practitioner's feedback. Although the synthesized DFRCM model is based on the two individual RCM models in Figure 4.9 and Figure 5.7 it requires further validation. One of the ways to carry out such validation could be through expert practitioners in the area of requirements and change management. In a similar fashion, implications for practice reported in this thesis are also planned to be validated by practitioners in the near future.

In addition, while this study was conducted in GSD settings the synthesized DFRCM model is currently limited in its coverage of the global dimensions and therefore needs further expansion. For instance, it could be extended (or merged into existing RCM models for GSD) to more fully represent the global software development environment by considering multiple sites and roles. The model is also extensible to include the artefacts and collaboration technologies relevant to the various lifecycle activities of a requirements change. Furthermore, in addition to seeking a better understanding of informal requirements change and its management process in global settings, its applicability in collocated development projects warrants investigation.

Informal requirements change is endemic to software development and imposes pressures on development teams, but it is an under-researched phenomenon. There is a need for further research to better understand this phenomenon and to develop suitable guidelines for practice. The research needs to recognize that informal requirements change serves a

necessary and useful purpose, and therefore it is essential to develop strategies and practices to accommodate its prevalence in practice.

The market-driven software aspects of this study have highlighted challenges similar to those reported by Carlshamre and Regnell (2000) which seem to drive informal requirements change. It is therefore necessary to find a palatable trade-off between requirements corresponding to perceived user needs and new, inventive ones that may provide a competitive advantage through the use of ground-breaking technology. Further research into market driven software development approaches and the management of informal requirements change could help address these challenges.

In this study a new perspective on the role and influence of the proxy client is highlighted which is at odds with the existing perception of the role in a GSD environment. This would benefit from further investigation and verification. Moreover, the proxy client role appeared to generate a large number of informal change requests – whether that is specific to this case or a more general deficiency in a proxy client boundary spanning role would be a fruitful area for research. Understanding this vital intermediary role and how it can contribute through the necessary interpretation of work across sites in a complex domain such as COTS based product evolution also presents an opportunity and warrants further investigation.

7.8 Chapter Summary

This final chapter of the thesis has reiterated the motivation for this study into the challenges of managing requirements change in GSD and the associated role of collaboration technologies along with its substantive and conceptual contributions. Managing requirements change in GSD settings is significantly different and difficult from that carried out in a collocated environment. The influence of distance (geographical, temporal and socio-cultural) hampers the possibility of managing change effectively. This chapter highlights the contemporary challenges faced by practitioners having to manage change in GSD environments where collaborative technology comes as a supporter of change management activities rather than an eliminator of challenges or a preventer from failures.

The evaluation of research presented has been provided to establish the credibility and reliability of this research. A series of recommendations for practice have been provided relating to requirements change management in the areas of packaged based COTS selection and customization, project management and change management. Some

recommendations have been made on the role and design of collaborative technologies in GSD environments.

For researchers likewise a set of recommendations for further research has been given. More research is recommended into the notion of informal requirements change management activities that are endemic to software development. An extension of the synthesized model of change management into a model applicable to globally distributed development environments is also recommended. Furthermore the role of a powerful proxy client and its implications on GSD projects has been identified as a fruitful area for future research.

References:

Ågerfalk, J., Fitzgerald, B., & In, O. P. (2006). Flexible and distributed software processes: old petunias in new bowls. In *Communications of the ACM*.

Akram, A., & Yasmeen, R. (2011). Attitudes towards English & Punjabi language learning in faisalabad. *Journal of Academic and Applied Studies, 1*(4), 9-32.

Ambler, S. W. (2014). *Agile Requirements Change Management*. Retrieved January 2, 2016, from http://agilemodeling.com/essays/changeManagement.htm

Ambler, S. W., & Lines, M. (2012). *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*: IBM Press.

Andres, H. P. (2013). Team cognition using collaborative technology: a behavioral analysis. *Journal of Managerial Psychology, 28*(1), 38-54.

Atlassian. (2015). *Confluence: Where Work Becomes Teamwork*. Retrieved September 11, 2015, from https://www.atlassian.com/software/confluence

Avison, D., & Fitzgerald, G. (2003). *Information systems development: methodologies, techniques and tools*: McGraw Hill.

Ayres, L. (2007). Qualitative research proposals-Part II: Conceptual models and methodological options. *Journal of Wound Ostomy & Continence Nursing, 34*(2), 131-133.

Bachrach, P., & Baratz, M. S. (1970). *Power and poverty: Theory and practice*: Oxford University Press.

Baldwin, J., & Damian, D. (2013). Tool usage within a globally distributed software development course and implications for teaching. Proceedings of the 3rd International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD), 2013.

Barbour, R. S. (2001). Checklists for improving rigour in qualitative research: a case of the tail wagging the dog? *BMJ: British Medical Journal, 322*(7294), 1115.

Bardram, J. E. (2009). Activity-based computing for medical work in hospitals. *ACM Transactions on Computer-Human Interaction (TOCHI), 16*(2), 10.

Barney, S., Wohlin, C., Chatzipetrou, P., & Angelis, L. (2011). Offshore insourcing: A case study on software quality alignment. Proceedings of the 6th IEEE International Conference on Global Software Engineering (ICGSE), 2011.

Bass, J. M., McDermott, R., & Lalchandani, J. T. (2015a, 13-16 July 2015). Virtual Teams and Employability in Global Software Engineering Education. Proceedings of the IEEE. 10th International Conference on Global Software Engineering (ICGSE), 2015. doi:10.1109/icgse.2015.21.

Bass, J. M., McDermott, R., & Lalchandani, J. T. (2015b, 13-16 July 2015). Virtual Teams and Employability in Global Software Engineering Education. Proceedings of the IEEE 10th International Conference on Global Software Engineering (ICGSE), 2015 doi:10.1109/icgse.2015.21

Beecham, S., OLeary, P., Richardson, I., Baker, S., & Noll, J. (2013). Who are we doing Global Software Engineering research for? IEEE. In proceedings of the 8th International Conference on Global Software Engineering (ICGSE), 2013

Bednar, P. M., & Welch, C. (2009). Contextual inquiry and requirements shaping. In *Information Systems Development* (pp. 225-236): Springer.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The case research strategy in studies of information systems. *MIS quarterly*, 369-386.

Benguria, G., Belén, A., Sellier, D., & Tay, S. (2002). European COTS User Working Group: Analysis of the Common Problems and Current Practices of the European COTS Users. In J. Dean & A. Gravel (Eds.), *COTS-Based Software Systems* (Vol. 2255, pp. 44-53): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-45588-4_5. doi:10.1007/3-540-45588-4_5

Berg, B. L., & Lune, H. (2011). *Qualitative research methods for the social sciences* (Vol. 5): Pearson Boston, MA.

Berlack, H. R. (1992). *Software configuration management*: Wiley Online Library.

Berry, D. M., Czarnecki, K., Antkiewicz, M., & AbdElRazik, M. (2010). Requirements determination is unstoppable: an experience report. Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'2010), 2010.

Bhat, J. M., Gupta, M., & Murthy, S. N. (2006). Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing. *Software, IEEE, 23*(5), 38-44. doi:10.1109/ms.2006.137

Biazzo, S. (2002). Process mapping techniques and organisational analysis: Lessons from sociotechnical system theory. *Business Process Management Journal, 8*(1), 42-52.

Bieg P, D. (2014). *Requirements Management — A Core Competency for Project and Program Success* Newtown Square, Pa 19073-3299 usa: Project Management Institute. Retrieved from http://www.pmi.org/~/media/PDF/Knowledge%20Center/PMI-Pulse-Requirements-Management-In-Depth-Report.ashx

Bird, C., Nagappan, N., Devanbu, P., Gall, H., & Murphy, B. (2009). Does distributed development affect software quality?: an empirical case study of Windows Vista. *Communications of the ACM, 52*(8), 85-93.

Birk, A., & Heller, G. (2015). *List of Requirements Management Tools*. Retrieved 22 October, 2015, from http://makingofsoftware.com/resources/list-of-rm-tools

Bjarnason, E., Wnuk, K., & Regnell, B. (2011). Requirements are slipping through the gaps—A case study on causes & effects of communication gaps in large-scale software development. Proceedings of the 19th International Requirements Engineering Conference (RE), 2011.

Bjorn, P., Bardram, J., Avram, G., Bannon, L., Boden, A., Redmiles, D.,Wulf, V. (2014 a). Global software development in a CSCW perspective. Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing.

Bjørn, P., Esbensen, M., Jensen, R. E., & Matthiesen, S. (2014 b). Does distance still matter? Revisiting the CSCW fundamentals on distributed collaboration. *ACM Transactions on Computer-Human Interaction, 21*(5), 1-27.

Boden, A., & Avram, G. (2009). Bridging knowledge distribution-The role of knowledge brokers in distributed software development teams. Proceedings of the ICSE Workshop on Cooperative and Human Aspects on Software Engineering, CHASE'09. 2009.

Boden, A., Avram, G., Bannon, L., & Wulf, V. (2009). Knowledge management in distributed software development teams-does culture matter? Proceedings of the Fourth IEEE International Conference on Global Software Engineering, 2009.

Bodker, S. (1997). Computers in mediated human activity. *Mind, culture, and activity, 4*(3), 149-158.

Boehm, B., & Abts, C. (1999). COTS integration: Plug and Pray? *Computer, 32*(1), 135-138.

Boehm, B., Grunbacher, P., & Briggs, R. O. (2001). Developing groupware for requirements negotiation: lessons learned. *Software, IEEE, 18*(3), 46-55.

Bohner, S. A. (1996). Software change impact analysis. *IEEE Computer Society Press.*

Bommer, M., DeLaPorte, R., & Higgins, J. (2002). Skunkworks approach to project management. *Journal of Management in Engineering, 18*(1), 21-28.

Booch, G., & Brown, A. W. (2003). Collaborative development environments. *Advances in computers, 59*, 1-27.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology, 3*(2), 77-101.

Braun, V., Clarke, V., Terry, G., Rohleder, P., & Lyons, A. (2014). Thematic analysis. *Qualitative Research in Clinical and Health Psychology*, 95.

Briand, L. C., Basili, V. R., Kim, Y. M., & Squier, D. R. (1994, September). A change analysis process to characterize software maintenance projects. In Software Maintenance, 1994. Proceedings., International Conference on (pp. 38-49). IEEE.

Brockmann, P. S., & Thaumüller, T. (2009). Cultural aspects of global requirements engineering: An empirical chinese-german case study. Proceedings of the Fourth International Conference on Global Software Engineering, 2009.

Bromley, D. B. (1986). *The case-study method in psychology and related disciplines*: John Wiley & Sons.

Brooks Jr, F. P. (1987). No silver bullet-essence and accidents of software engineering. *IEEE computer, 20*(4), 10-19.

Brooks, R. (1991, August). Comparative task analysis: An alternative direction for human-computer interaction science. In *Designing interaction* (pp. 50-59). Cambridge University Press.

Buchan, J., Ekadharmawan, C. H., & MacDonell, S. G. (2009). Insights into Domain Knowledge Sharing in Software Development Practice in SMEs. 93-100. doi:10.1109/apsec.2009.47

Cairncross, F. (1997). *The Death of Distance: How the Communications Revolution Will Change Our Lives*: Harvard Business School Press. Retrieved from https://books.google.co.nz/books?id=d6Xk9wOH_3IC

Cajander, A., Daniels, M., Cullhed, M., Clear, T., McDermott, R., & Laxer, C. (2012, October). Categorizing how students use Collaborative Technologies in a globally distributed project. In *Frontiers in Education Conference (FIE), 2012* (pp. 1-6). IEEE.

Calefato, F., Damian, D., & Lanubile, F. (2012). Computer-mediated communication to support distributed requirements elicitations and negotiations tasks. *Empirical Software Engineering, 17*(6), 640-674.

Carlshamre, P., & Regnell, B. (2000). Requirements lifecycle management and release planning in market-driven requirements engineering processes. Proceedings of the 11th International Workshop on Database and Expert Systems Applications, 2000.

Carmel, E. (1999). *Global software teams: collaborating across borders and time zones*: Prentice Hall PTR.

Carmel, E., & Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. *Software, IEEE, 18*(2), 22-29.

Carmel, E., Espinosa, J. A., & Dubinsky, Y. (2010). " Follow the Sun" Workflow in Global Software Development. *Journal of Management Information Systems, 27*(1), 17-38.

Carmel, E., & Kojola, E. (2012). Timeshifting into the Night: Guidelines vs. Practices Affecting Time Zone Dependent Workers. (September 17, 2012).
Carrillo, d. G. J. M., Nicolás, J., Fernández, A. J. L., Toval, A., Ebert, C., & Vizcaíno, A. (2015). Commonalities and differences between requirements engineering tools: A quantitative approach. *Computer Science and Information Systems,*(12(1)), 257-288.
Carter, R., Antón, A., Dagnino, A., & Williams, L. (2001). Evolving beyond requirements creep: a risk-based evolutionary prototyping model. Proceedings. Fifth International Symposium on Requirements Engineering, 2001.
Checkland, P. (1999). *Systems thinking, systems practice: includes a 30-year retrospective*: John Wiley and Sons Ltd.
Cheng, B. H. C., & Atlee, J. M. (2007). *Research Directions in Requirements Engineering*. 2007 Future of Software Engineering, doi:10.1109/fose.2007.17
Cheng, L.-T., Hupfer, S., Ross, S., & Patterson, J. (2003). *Jazzing up Eclipse with collaborative tools*. Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange, Anaheim, California. doi:10.1145/965660.965670
Clear, T., & MacDonell, S. G. (2011). Understanding technology use in global virtual teams: Research methodologies and methods. *Information and Software Technology, 53*(9), 994-1011. doi:10.1016/j.infsof.2011.01.011
Clear, T., Raza, B., & MacDonell, S. G. (2013). A Critical Evaluation of Failure in a Nearshore Outsourcing Project: What Dilemma Analysis Can Tell Us. Proceedings of the IEEE 8th International Conference on Global Software Engineering (ICGSE), 2013.
Cleland-Huang, J., Chang, C. K., & Christensen, M. (2003). Event-based traceability for managing evolutionary change. *Software Engineering, IEEE Transactions on, 29*(9), 796-810.
Cleland-Huang, J., & Damian, D. (2011). Ready-set-transfer! Technology transfer in the requirements engineering domain. Proceedings of IEEE 19th International the Requirements Engineering Conference (RE), 2011.
Cleland-Huang, J., Gotel, O., & Zisman, A. (2012). *Software and systems traceability* (Vol. 2): Springer.
CMMI Product Team. (2006). CMMI for Development, version 1.2.
Cockburn, A. (2003). *People and Methodologies in Software Development* (Thesis). University of Oslo.
Cook, T. D., Campbell, D. T., & Day, A. (1979). *Quasi-experimentation: Design & analysis issues for field settings*: Houghton Mifflin Boston.
Costello, R. J., & Liu, D.-B. (1995). Metrics for requirements engineering. *Journal of systems and software, 29*(1), 39-63.
Cramton, C. D. (2001). The mutual knowledge problem and its consequences for dispersed collaboration. *Organization science, 12*(3), 346-371.
Creswell, J. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches*: SAGE Publications, Incorporated.
Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*: Sage.
Creswell, J. W., & Miller, D. L. (2000). Determining validity in qualitative inquiry. *Theory into practice, 39*(3), 124-130.
D'Cruz, P., & Noronha, E. (2015). Ambivalence: Employee responses to depersonalized bullying at work. *Economic and Industrial Democracy, 36*(1), 123-145.
Dalcher, D. (2014). Who Needs Project Requirements? *PM World Journal*.
Damian, D. (2007). Stakeholders in global requirements engineering: Lessons learned from practice. *Software, IEEE, 24*(2), 21-27.
Damian, D., & Chisan, J. (2006). An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. *Software Engineering, IEEE Transactions on, 32*(7), 433-453.
Damian, D., Chisan, J., Allen, P., & Corrie, B. (2003, May). Awareness meets requirements management: awareness needs in global software development. In *Proc. of the International Workshop on Global Software Development, International Conference on Software Engineering* (pp. 7-11).
Damian, D., Chisan, J., Vaidyanathasamy, L., & Pal, Y. (2005). Requirements engineering and downstream software development: Findings from a case study. *Empirical Software Engineering, 10*(3), 255-283.
Damian, D., Marczak, S., Kwan, I., & Ebert, C. (2012). Practice: Requirements Engineering in Global Teams. *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing*, 257-267.
Damian, D., & Moitra, D. (2006). Guest Editors' Introduction: Global Software Development: How Far Have We Come? *Software, IEEE, 23*(5), 17-19.
Damian, D., & Zowghi, D. (2003). An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.

Damian, D., & Zowghi, D. (2003b). RE challenges in multi-site software development organisations. *Requirements Engineering, 8*(3), 149-160.

Damian, D. E., & Zowghi, D. (2002, September). The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on* (pp. 319-328). IEEE.

Darke, P., Shanks, G., & Broadbent, M. (1998). Successfully completing case study research: combining rigour, relevance and pragmatism. *Information systems journal, 8*(4), 273-289.

Davies, L., & Nielsen, S. (1992). An ethnographic study of configuration management and documentation practices. Proceedings of the IFIP WG8. 2 Working Conference on The Impact of Computer Supported Technologies in Information Systems Development

Davis, A. M. (2013). *Just enough requirements management: where software development meets marketing*. New York: Addison-Wesley.

Davis, A. M., & Hickey, A. M. (2009). The Role of China in Requirements Research. In *The China Information Technology Handbook* (pp. 1-28): Springer.

Davis, A. M., & Nori, K. V. (2007). Requirements, Plato's Cave, and Perceptions of Reality. Proceedings of the 31st Annual International Computer Software and Applications Conference, (COMPSAC) 2007.

Davis, A., Nurmuliani, N., Park, S., & Zowghi, D. (2008). Requirements Change: What's the Alternative?. In *International Computer Software and Applications Conference*. IEEE Explore.

de Gea, J. M., Nicolas, J., Aleman, J. L. F., Toval, A., Ebert, C., & Vizcaino, A. (2011). Requirements Engineering Tools. *Software, IEEE, 28*(4), 86-91. doi:10.1109/ms.2011.81

de Gea, J. M., Nicolás, J., Alemán, J. L. F., Toval, A., Vizcaíno, A., & Ebert, C. (2013). Reusing requirements in global software engineering. In *Managing requirements knowledge* (pp. 171-197): Springer.

de Gea, J. M., Nicolás, J., Fernández, A. J. L., Toval, A., Ebert, C., & Vizcaíno, A. (2015). Commonalities and differences between requirements engineering tools: A quantitative approach. *Computer Science and Information Systems*(00), 1-1.

de Gea, J. M., Nicolás, J., Fernández Alemán, J. L., Toval, A., Ebert, C., & Vizcaíno, A. (2012). Requirements engineering tools: Capabilities, survey and assessment. *Information and Software Technology, 54*(10), 1142-1157.

de Souza, C. R., & Redmiles, D. F. (2003). Opportunities for extending activity theory for studying collaborative software development. Proceedings of the In Workshop on Applying Activity Theory to CSCW Research and Practice, in conjunction with ECSCW

Derntl, M., Renzel, D., Nicolaescu, P., Koren, I., & Klamma, R. (2015, 13-16 July 2015). Distributed Software Engineering in Collaborative Research Projects. Proceedings of the Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference on doi:10.1109/ICGSE.2015.12

DeSanctis, G., & Poole, M. S. (1994). Capturing the complexity in advanced technology use: Adaptive structuration theory. *Organization science*, 121-147.

DeSantis, L., & Ugarriza, D. N. (2000). The concept of theme as used in qualitative nursing research. *Western Journal of Nursing Research, 22*(3), 351-372.

Deshpande, S., Beecham, S., & Richardson, I. (2011). Global Software Development Coordination Strategies - A Vendor Perspective. In J. Kotlarsky, L. Willcocks, & I. Oshri (Eds.), *New Studies in Global IT and Business Service Outsourcing* (Vol. 91, pp. 153-174): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-24815-3_9. doi:10.1007/978-3-642-24815-3_9

Deshpande, S., Richardson, I., Casey, V., & Beecham, S. (2010). Culture in global software development-a weakness or strength? Proceedings of the 5th IEEE International Conference on Global Software Engineering (ICGSE), 2010

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of systems and software, 85*(6), 1213-1221. doi:10.1016/j.jss.2012.02.033

Dittrich, Y., Vaucouleur, S., & Giff, S. (2009). ERP customization as software engineering: knowledge sharing and cooperation. *Software, IEEE, 26*(6), 41-47.

Dourish, P., & Bellotti, V. (1992, December). Awareness and coordination in shared workspaces. In Proceedings of the 1992 ACM conference on Computer-supported cooperative work (pp. 107-114). ACM.

Dutoit, A. H., McCall, R., Mistrík, I., & Paech, B. (2006). Rationale management in software engineering: Concepts and techniques. In *Rationale management in software engineering* (pp. 1-48): Springer.

Dutoit, A. H., & Paech, B. (2003). Eliciting and maintaining knowledge for requirements evolution. In *Managing software engineering knowledge* (pp. 135-155): Springer.

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering* (pp. 285-311): Springer.
Ebert, C. (2005). Requirements before the requirements: understanding the upstream impacts. Proceedings of the 13th IEEE International Conference on Requirements Engineering, 2005.
Ebert, C. (2011). The dark side: challenges. *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing*, 19-25.
Ebert, C. (2014). Managing Global Software Projects. In G. Ruhe & C. Wohlin (Eds.), *Software Project Management in a Changing World* (pp. 223-246): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-55035-5_9. doi:10.1007/978-3-642-55035-5_9
Ebert, C., & De Man, J. (2005). Requirements uncertainty: influencing factors and concrete improvements. Proceedings of the 27th international conference on Software engineering
Ebert, C., & De Neve, P. (2001). Surviving global software development. *Software, IEEE, 18*(2), 62-69.
Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of management review, 14*(4), 532-550.
Esbensen, M., Tell, P., Cholewa, J. B., Pedersen, M. K., & Bardram, J. (2015). The dBoard: A Digital Scrum Board for Distributed Software Development. Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces
Espinosa, J. A., & Carmel, E. (2004, January). The effect of time separation on coordination costs in global software teams: A dyad model. In *System Sciences, 2004.* Proceedings of the 37th Annual Hawaii International Conference on (pp. 10-pp). IEEE.
Espinosa, J. A., Cummings, J. N., Wilson, J. M., & Pearce, B. M. (2003). Team boundary issues across multiple global firms. *Journal of Management Information Systems, 19*(4), 157-190.
Espinoza, A., & Garbajosa, J. (2011). A study to support agile methods more effectively through traceability. *Innovations in Systems and Software Engineering, 7*(1), 53-69.
Estublier, J., Leblang, D., Hoek, A., Conradi, R., Clemm, G., Tichy, W., & Wiborg-Weber, D. (2005). Impact of software engineering research on the practice of software configuration management. *ACM Transactions on Software Engineering and Methodology (TOSEM), 14*(4), 383-430.
Fauzi, S. S. M., Bannerman, P. L., & Staples, M. (2010, November). Software Configuration Management in Global Software Development: A Systematic Map. In *Software Engineering Conference (APSEC), 2010 17th Asia Pacific* (pp. 404-413). IEEE.
FBO.Gov. (2011). Project Solicitation Offer and Award (pp. 54).
FBO.Gov. (2011a). Draft RFPs/RFIs, Responses to Questions (pp. 54).
Fitzgerald, B., & Howcroft, D. (1998). Towards dissolution of the IS research debate: from polarization to polarity. *Journal of Information Technology, 13*, 313-326.
Flanagan, J. C. (1954). The critical incident technique. *Psychological bulletin, 51*(4), 327.
Flyvbjerg, B. (2006). Five misunderstandings about case-study research. *Qualitative inquiry, 12*(2), 219-245.
Fowler, M. (2006). *Using an agile software process with offshore development*. Retrieved January 2, 2016,
Friedman, R. A., & Currall, S. C. (2003). Conflict escalation: Dispute exacerbating elements of e-mail communication. *Human relations, 56*(11), 1325-1347.
Gable, G. G. (1994). Integrating case study and survey research methods: an example in information systems. *European Journal of Information Systems, 3*(2), 112-126.
Gallivan, M. J., & Oh, W. (1999). Analyzing IT outsourcing relationships as alliances among multiple clients and vendors. Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences HICSS-32, 1999.
Gattiker, T. F., & Goodhue, D. L. (2000). Understanding the plant level costs and benefits of ERP: will the ugly duckling always turn into a swan? Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.
George, O., Owoyemi, O., & Onakala, U. (2012). Hofstede's 'Software of the Mind'Revisited and Tested: The Case of Cadbury Worldwide and Cadbury (Nigeria) Plc-A Qualitative Study. *International Business Research, 5*(9), p148.
Ghaisas, S., & Ajmeri, N. (2013). Knowledge-assisted ontology-based requirements evolution. In *Managing requirements knowledge* (pp. 143-167): Springer.
Gibbert, M., Ruigrok, W., & Wicki, B. (2008). What passes as a rigorous case study? *Strategic management journal, 29*(13), 1465-1474.
Gibbons, M. T. (1987). Introduction: The politics of interpretation. *Interpreting politics*, 1-31.
Gibson, W., & Brown, A. (2009). *Working with qualitative data* (illustrated ed.): SAGE Publications Limited.
Gillham, B. (2005). *Research Interviewing: The range of techniques: A practical guide*: McGraw-Hill Education (UK).

Glinz, M., & Fricker, S. (2013). On Shared Understanding in Software Engineering. Proceedings of the Software Engineering.
Glinz, M., & Fricker, S. A. (2014). On shared understanding in software engineering: an essay. *Computer Science-Research and Development*, 1-14.
Gorschek, T., Fricker, S., Palm, K., & Kunsman, S. A. (2010). A lightweight innovation process for software-intensive product development. *IEEE software, 27*(1), 37.
Gorschek, T., Gomes, A., Pettersson, A., & Torkar, R. (2012). Introduction of a process maturity model for market-driven product management and requirements engineering. *Journal of Software: Evolution and Process, 24*(1), 83-113.
Green, J., & Thorogood, N. (2013). *Qualitative methods for health research*. London: SAGE Publications.
Green, J. L., Camilli, G., & Elmore, P. B. (2012). *Handbook of complementary methods in education research*. Washington: Lawrence Erlbaum Associate, Publishers.
Gregory, R., Beck, R., & Prifling, M. (2009). Breaching the knowledge transfer blockade in IT offshore outsourcing projects-A case from the financial services industry. Proceedings of the 42nd Hawaii International Conference onSystem Sciences HICSS'09, 2009.
Grehag, Å. (2001, June). Requirements management in a life-cycle perspective-A position paper. In *Proceedings of the 7th International Workshop on REFSQ* (pp. 183-188).
Greiner, L. E. (1989). Evolution and Revolution as Organizations Grow [Greiner1989]. In D. Asch & C. Bowman (Eds.), *Readings in Strategic Management* (pp. 373-387). London: Macmillan Education UK. Retrieved from http://dx.doi.org/10.1007/978-1-349-20317-8_25. doi:10.1007/978-1-349-20317-8_25
GSD Inc. (2009). Requirements Management Process Definition Version 1.2.
GSD Inc. (2011). Task A1 Report 7th Sept
GSD Inc. (2012). Stakeholder Committment Sheet.
GSD Inc. (2012a). Requirements Management and Traceability Matrix Workbook
GSD Inc. (2012c). Software Design Document Task A2.
GSD Inc. (2012e). Project Categorization for Customization Document 22nd May.
GSD Inc. (2012f). CMMI documentation Nov.
GSD Inc. (2012g). Task A-1 Report.
GSD Inc. (2013a). Administration Site Design 15th March
GSD Inc. (2013b). Design Document List 15 March
Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. *Handbook of qualitative research, 2*, 163-194.
Hall, R. (2002). Enterprise resource planning systems and organizational change: transforming work organization?. Strategic Change, 11(5), 263-270.
Halverson, C. A., Ellis, J. B., Danis, C., & Kellogg, W. A. (2006, November). Designing task visualizations to support the coordination of work in software development. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (pp. 39-48). ACM.
Hanisch, J., & Corbitt, B. (2007). Impediments to requirements engineering during global software development. *European Journal of Information Systems, 16*(6), 793-805.
Hansen, S., Kharabe, A., & Lyytinen, K. (2013). The Structures of Computation: A Distributed Cognitive Analysis of Requirements Evolution in Diverse Software Development Environments. Proceedings of the 8th Pre-ICIS International Research Workshop on Information Technology Project Management (IRWITPM), 2013.
Hanson, J. H., & Brophy, P. D. (2012). The Critical Incident Technique: An Effective Tool For Gathering Experience From Practicing Engineers. *Advances in Engineering Education, 3*(1).
Harjani, D.-R., & Queille, J.-P. (1992). A process model for the maintenance of large space systems software. Proceedings of the Software Maintenance, 1992. Proceerdings., Conference on
Hass, A. M. J. (2003). *Configuration management principles and practice*: Addison-Wesley Professional.
Hatch, J. A. (2002). *Doing qualitative research in education settings*: SUNY Press.
Heindl, M., Reinisch, F., & Biffl, S. (2007). Requirements Management Infrastructures in Global Software Development. Proceedings of the international conference on global software engineering (ICGSE), workshop on tool-supported requirements management in distributed projects (REMIDI), Munich, Germany.
Herbsleb, J. D. (2007, May). Global software engineering: The future of socio-technical coordination. In *2007 Future of Software Engineering* (pp. 188-198). IEEE Computer Society.
Herbsleb, J. D., & Grinter, R. E. (1999, May). Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of the 21st international conference on Software engineering* (pp. 85-95). ACM.
Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on, 29*(6), 481-494.

Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2001, July). An empirical study of global software development: distance and speed. In *Proceedings of the 23rd international conference on software engineering* (pp. 81-90). IEEE Computer Society.

Herriott, R. E., & Firestone, W. A. (1983). Multisite qualitative policy research: Optimizing description and generalizability. *Educational researcher*, 14-19.

Hester, A. J. (2010). Increasing collaborative knowledge management in your organization: characteristics of wiki technology and wiki users. Proceedings of the 2010 Special Interest Group on Management Information System's 48th annual conference on Computer personnel research on Computer personnel research

Hiisilä, H., Kauppinen, M., & Kujala, S. (2015). Challenges of the Customer Organization's Requirements Engineering Process in the Outsourced Environment–A Case Study. In *Requirements Engineering: Foundation for Software Quality* (pp. 214-229): Springer.

Hofstede, G. H., & Hofstede, G. (2001). *Culture's consequences: Comparing values, behaviors, institutions and organizations across nations*: Sage.

Hong, K.-K., & Kim, Y.-G. (2002). The critical success factors for ERP implementation: an organizational fit perspective. *Information & Management, 40*(1), 25-40.

Hossain, E., Babar, M. A., & Paik, H.-y. (2009). Using scrum in global software development: a systematic literature review. Proceedings of the Fourth IEEE International Conference on Global Software Engineering ICGSE, 2009.

Hsieh, Y. (2006). Culture and shared understanding in distributed requirements engineering. Proceedings of the International Conference on Global Software Engineering ICGSE'06, 2006.

Huberman, M., & Miles, M. B. (2002). *The qualitative researcher's companion*: California Sage Publications.

Hull, E., Jackson, K., & Dick, J. (2010). Management Aspects of Requirements Engineering. 159-180. doi:10.1007/978-1-84996-405-0_8

Hurtado Alegría, J. A., Bastarrica, M. C., Quispe, A., & Ochoa, S. F. (2014). MDE-based process tailoring strategy. *Journal of Software: Evolution and Process, 26*(4), 386-403.

Hussain, W., Buchan, J., & Clear, T. (2014). Managing Requirements in Globally Distributed COTS Customization. Proceedings of the IEEE International Conference on Global Software Engineeering Workshops (ICGSEW), 2014 doi:10.1109/icgsew.2014.13

Hussain, W., & Clear, T. (2012). GRCM: a model for Global Requirements Change Management *2nd International Requirements Engineering Efficiency Workshop (REEW 2012)*. Essen, Germany.

Hussain, W., & Clear, T. (2014). Spreadsheets as collaborative technologies in global requirements change management. Proceedings of the 9th International Conference on Global Software Engineering (ICGSE) 2014.

ICAC. (2013). *Managing IT contractors, improving IT outcomes*. Sydney, Australia. : Independent Commission Against Corruption.

Imtiaz, S., Ikram, N., & Imtiaz, S. (2008). A process model for managing requirement change Proceedings of the Fourth IASTED International Conference on Advances in Computer Science and Technology, 2008.

Jaanu, T., Paasivaara, M., & Lassenius, C. (2012). Effects of four distances on communication processes in global software projects. Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2012

Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., & Robinson, W. (2010, June). The brave new world of design requirements: Four key principles. In Advanced Information Systems Engineering (pp. 470-482). Springer Berlin Heidelberg.

Johansson, B., & de Carvalho, R. A. (2010). Software tools for requirements management in an ERP system context. Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 169-170). ACM.

Jones, C. (1996). Strategies for managing requirements creep. *Computer, 29*(6), 92-94.

Jørgensen, M. (2014). Failure factors of small software projects at a global outsourcing marketplace. *Journal of systems and software, 92*, 157-169.

Kääriäinen, J., & Välimäki, A. (2009). Applying application lifecycle management for the development of complex systems: Experiences from the automation industry. In *Software Process Improvement* (pp. 149-160): Springer.

Kanaracus, C. (2014). *Global IT spending outlook 'better but subpar' for 2014, Forrester says*. Retrieved 24 October, 2015, from http://www.computerworld.com/article/2487254/it-management/global-it-spending-outlook--better-but-subpar--for-2014--forrester-says.html

Kaplan, B., & Duchon, D. (1988). Combining qualitative and quantitative methods in information systems research: a case study. *MIS quarterly*, 571-586.

Karhu, A. (2011). *Improving the usability of wikis: Case IT product team of Company X*. Master Thesis, Aalto University, (12509), Finland.

Karlsson, L., Dahlstedt, Å. G., Regnell, B., Natt och Dag, J., & Persson, A. (2007). Requirements engineering challenges in market-driven software development–An interview study with practitioners. *Information and Software Technology, 49*(6), 588-604.

Keil, P., Kuhrmann, M., & Niinimaki, T. (2011, 15-18 Aug. 2011). 5th International Workshop on Tool Support Development and Management in Distributed Software Projects (REMIDI'11). Proceedings of the Sixth IEEE International Conference on Global Software Engineering Workshop (ICGSEW), 2011 doi:10.1109/icgse-w.2011.25

Keith, M., Demirkan, H., & Goul, M. (2009). Understanding Coordination in IT Project-Based Environments: An Examination of Team Cognition and Virtual Team Efficacy. Proceedings of the 42nd Hawaii International Conference on System Sciences, HICSS'09, 2009.

Kelanti, Hyysalo J, Välimäki A, Kuvaja P, & M, O. (2013, October 27). A Case Study of Requirements Management: Toward Transparency in Requirements Management Tools. Proceedings of the ICSEA 2013 : The Eighth International Conference on Software Engineering Advances Venice, Italy.

Kellogg, W. A. (1990, August). Qualitative artifact analysis. In *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction* (pp. 193-198). North-Holland Publishing Co.

Kien, S. S., & Soh, C. (2003). An Exploratory Analysis of the Sources and Nature of Misfits in ERP Implementations. *Second-Wave Enterprise Resource Planning Systems*, 373.

Kiniti, S., & Standing, C. (2013). Wikis as knowledge management systems: issues and challenges. *Journal of Systems and Information Technology, 15*(2), 189-201.

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, 67-93.

Klitmøller, A., & Lauring, J. (2013). When global virtual teams share knowledge: Media richness, cultural difference and language commonality. *Journal of World Business, 48*(3), 398-406. doi:10.1016/j.jwb.2012.07.023

Knauss, E., Damian, D., Cleland-Huang, J., & Helms, R. (2014). Patterns of continuous requirements clarification. *Requirements Engineering*, 1-21. doi:10.1007/s00766-014-0205-z

Kock Jr, N. F., McQueen, R. J., & John, L. S. (1997). Can action research be made more rigorous in a positivist sense? The contribution of an iterative approach.Journal of Systems and Information Technology,1(1), 1-23.

Kommeren, R., & Parviainen, P. (2007). Philips experiences in global distributed software development. *Empirical Software Engineering, 12*(6), 647-660. doi:10.1007/s10664-007-9047-3

Korpela, M., Mursu, A., & Soriyan, H. A. (2002). Information systems development as an activity. *Computer Supported Cooperative Work (CSCW), 11*(1-2), 111-128.

Kowark, T., & Plattner, H. (2012, January). AnalyzeD: a shared tool for analyzing virtual team collaboration in classroom software engineering projects. In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)* (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). Krauss, S. E. (2005). Research paradigms and meaning making: A primer. *The qualitative report, 10*(4), 758-770.

Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. Communications of the ACM, 38(3), 69-82.

Kroll, J., Hashmi, S. I., Richardson, I., & Audy, J. L. (2013, August). A systematic literature review of best practices and challenges in follow-the-sun software development. In *Global Software Engineering Workshops (ICGSEW), 2013 IEEE 8th International Conference on* (pp. 18-23). IEEE.

Kuhrmann, M., & Kalus, G. (2008). Providing integrated development processes for distributed development environments. Proceedings of the Workshop on Supporting Distributed Team Work at Computer Supported Cooperative Work (CSCW 2008)

Lai, R., & Ali, N. (2013). A Requirements Management Method for Global Software Development. *AIS: Advances in Information Sciences, 1*, 38-58.

Lam, W., Loomes, M., & Shankararaman, V. (1999). Managing requirements change using metrics and action planning. In *Software Maintenance and Reengineering, 1999. Proceedings of the Third European Conference on* (pp. 122-128). IEEE. doi:10.1109/csmr.1999.756689

Lang, M., & Duggan, J. (2001). A tool to support collaborative software requirements management. *Requirements Engineering, 6*(3), 161-172.

Lanubile, F. (2009). Collaboration in distributed software development. In *Software Engineering* (pp. 174-193): Springer.

Lanubile, F., Calefato, F., & Ebert, C. (2013). Group Awareness in Global Software Engineering. *IEEE software, 30*(2), 18-23.

Lanubile, F., Ebert, C., Prikladnicki, R., & Vizcaíno, A. (2010). Collaboration tools for global software engineering. *Software, IEEE, 27*(2), 52-55.

Larsson, S. (2009). A pluralist view of generalization in qualitative research. *International journal of research & method in education, 32*(1), 25-38.
Lauesen, S. (2006). COTS tenders and integration requirements. *Requirements Engineering, 11*(2), 111-122.
Laurent, P., Mader, P., Cleland-Huang, J., & Steele, A. (2010, August). A taxonomy and visual notation for modeling globally distributed requirements engineering projects. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on* (pp. 35-44). IEEE.
Leffingwell, D., & Widrig, D. (2003). *Managing software requirements: a use case approach*: Pearson Education.
Lehman, M. (1998). Software's future: Managing evolution. *IEEE software*(1), 40-44.
Lethbridge, T. C., Sim, S. E., & Singer, J. (2005). Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering, 10*(3), 311-341.
Levin, G., & Ward, J. L. (2011). *Program management complexity: A competency model*. Florida: CRC Press.
Lincoln, Y. S., & Guba, E. G. (1990). Judging the quality of case study reports. *Internation Journal of Qualitative Studies in Education, 3*(1), 53-59.
Lings, B., Lundell, B., Ågerfalk, P. J., & Fitzgerald, B. (2007, August). A reference model for successful Distributed Development of Software Systems. In *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on* (pp. 130-139). IEEE.
Livermore, J. A. (2008). Factors that significantly impact the implementation of an agile software development methodology. *Journal of Software, 3*(4), 31-36.
Lopez, A., Nicolas, J., & Tov, A. (2009, July). Risks and safeguards for the requirements engineering process in global software development. In *Global Software Engineering, 2009.* ICGSE 2009. Fourth IEEE International Conference on (pp. 394-399). IEEE.
Ma, L., Nuseibeh, B., Piwek, P., De Roeck, A., & Willis, A. (2009, September). On presuppositions in requirements. In *Managing Requirements Knowledge (MARK), 2009* Second International Workshop on (pp. 27-31). IEEE.
Maalej, W., Thurimella, A. K., Happel, H. J., & Decker, B. (2008, September). Managing Requirements Knowledge (MaRK_08). In *Managing Requirements Knowledge, 2008.* MARK'08. First International Workshop on (pp. i-ii). IEEE.
Mackenzie, N., & Knipe, S. (2006). Research dilemmas: Paradigms, methods and methodology. *Issues in educational research, 16*(2), 193-205.
Malone, T. W., Lai, K.-Y., & Fry, C. (1995). Experiments with Oval: a radically tailorable tool for cooperative work. *ACM Transactions on Information Systems (TOIS), 13*(2), 177-205.
Manning, P. K., & Cullum-Swan, B. (1994). Narrative, content, and semiotic analysis. *Handbook of qualitative research*, 463-477.
Manteli, C., Van Den Hooff, B., Tang, A., & Van Vliet, H. (2011, August). The impact of multi-site software governance on knowledge management. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on* (pp. 40-49). IEEE.
Massey, A. P., Hung, Y.-T. C., Montoya-Weiss, M., & Ramesh, V. (2001). When culture and style aren't about clothes: perceptions of task-technology fit in global virtual teams. Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work
Shaw, M. (2003, May). Writing good software engineering research papers: minitutorial. In Proceedings of the 25th international conference on software engineering (pp. 726-736). IEEE Computer Society.
McGee, S., & Greer, D. (2010). Sources of software requirements change from the perspectives of development and maintenance. *International Journal on Advances in Software, 3*(1 and 2), 186-200.
McGee, S., & Greer, D. (2012). Towards an understanding of the causes and effects of software requirements change: two case studies. *Requirements Engineering, 17*(2), 133-155. doi:10.1007/s00766-012-0149-0
McGrath, J. E., & Brinberg, D. (1983). External validity and the research process: A comment on the Calder/Lynch dialogue. *Journal of Consumer Research*, 115-124.
Mertens, D. M. (2005). *Research methods in education and psychology: Integrating diversity with quantitative & qualitative approaches* (2nd ed. ed.): Thousand Oaks: Sage.
Meso, P., & Jain, R. (2006). Agile software development: adaptive systems principles and best practices. *Information Systems Management, 23*(3), 19-30.
Miles, M. B., & Huberman, A. M. (1984). Drawing valid meaning from qualitative data: Toward a shared craft. *Educational researcher, 13*(5), 20-30.
Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*: California, Sage Publications.

Mishra, D., & Alok, M. (2011). Research trends in management issues of global software development: evaluating the past to envision the future. *Journal of Global Information Technology Management, 14*(4), 48-69.

Mistrík, I., Grundy, J., van der Hoek, A., & Whitehead, J. (2010). *Collaborative Software Engineering*: Springer. Retrieved from http://books.google.co.nz/books?id=NOprLdi25qUC

Mockus, A., & Herbsleb, J. (2001). Challenges of global software development. In *Software Metrics Symposium, 2001. METRICS 2001. Seventh International* (pp. 182-184). doi:10.1109/metric.2001.915526

Mohamed, A., Ruhe, G., & Eberlein, A. (2007, March). COTS selection: past, present, and future. In *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the* (pp. 103-114). IEEE.

Mohamed, A., Ruhe, G., & Eberlein, A. (2007). MiHOS: an approach to support handling the mismatches between system requirements and COTS products. *Requirements Engineering*, *12*(3), 127-143.

Mohan, K., Xu, P., Cao, L., & Ramesh, B. (2008). Improving change management in software development: Integrating traceability and software configuration management. *Decision Support Systems, 45*(4), 922-936.

Monasor, M. J., Vizcaíno, A., & Piattini, M. (2010, April). A training tool for Global Software Development. In *Information Technology Based Higher Education and Training (ITHET), 2010 9th International Conference on* (pp. 9-16). IEEE.

Monteiro, M. R., Ebert, C., & Recknagel, M. (2009, August). Improving the exchange of requirements and specifications between business partners. In *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International* (pp. 253-260). IEEE.

Moore, R., Reff, K., Graham, J., & Hackerson, B. (2007). Scrum at a fortune 500 manufacturing company. Proceedings of the Agile Conference (AGILE), 2007

Morisio, M., Seaman, C. B., Parra, A. T., Basili, V. R., Kraft, S. E., & Condon, S. E. (2000, June). Investigating and improving a COTS-based software development. In *Proceedings of the 22nd international conference on Software engineering* (pp. 32-41). ACM.

Mumford, E. (1995). *Effective systems design and requirements analysis: the ETHICS approach*: Macmillan.

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and organization, 17*(1), 2-26.

Nardi, B. A. (1996). Studying context: A comparison of activity theory, situated action models, and distributed cognition. *Context and consciousness: Activity theory and human-computer interaction*, 69-102.

Nardi, B. A., & Miller, J. R. (1991). Twinkling lights and nested loops: distributed problem solving and spreadsheet development. *International Journal of Man-Machine Studies, 34*(2), 161-184.

Nazir, S., Tayyab, A., Sajid, A., Rashid, H. U., & Javed, I. (2012). How Online Shopping Is Affecting Consumers Buying Behavior in Pakistan. *International Journal of Computer Science Issues, 9*(3), 486-495.

Neale, D. C., Carroll, J. M., & Rosson, M. B. (2004). Evaluating computer-supported cooperative work: models and frameworks. Proceedings of the 2004 ACM conference on Computer supported cooperative work

Newcomb, T. M. (1961). The acquaintance process: Holt, Rinehart and Winston. Retrieved from https://books.google.co.nz/books?id=YO4EAAAAMAAJ

Niazi, M., Hickman, C., Ahmad, R., & Ali Babar, M. (2008). A model for requirements change management: Implementation of CMMI level 2 specific practice. *Product-Focused Software Process Improvement*, 143-157.

Niinimäki, T., Piri, A., Lassenius, C., & Paasivaara, M. (2010, August). Reflecting the choice and usage of communication tools in GSD projects with media synchronicity theory. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on* (pp. 3-12). IEEE.

Nolan, A. J., Abrahão, S., Clements, P., & Pickard, A. (2011, August). Managing requirements uncertainty in engine control systems development. In the *19th IEEE International Requirements Engineering Conference (RE), 2011* (pp. 259-264).

Noll, J., Beecham, S., & Richardson, I. (2010). Global software development and collaboration: barriers and solutions. *ACM Inroads, 1*(3), 66-78.

Nurdiani, I., Jabangwe, R., Smite, D., & Damian, D. (2011, August). Risk identification and risk mitigation instruments for global software development: Systematic review and survey results. In *Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on* (pp. 36-41). IEEE.

Nurmuliani, N., Zowghi, D., & Powell, S. (2004). Analysis of requirements volatility during software development life cycle. In *Software Engineering Conference, 2004. Proceedings. 2004 Australian* (pp. 28-37). IEEE. Retrieved from

http://ieeexplore.ieee.org/ielx5/9061/28748/01290455.pdf?tp=&arnumber=1290455&isnumber=28748 doi:10.1109/aswec.2004.1290455
Nuseibeh, B., & Easterbrook, S. (2000, May). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 35-46). ACM.
Ó Conchúir, E. (2010). Global software development: a multiple-case study of the realisation of the benefits. Doctoral Thesis. University of Limerikc. Retrieved from https://ulir.ul.ie/handle/10344/435
O'Sullivan, P. B., & Flanagin, A. J. (2003). Reconceptualizing 'flaming'and other problematic messages. *New Media & Society, 5*(1), 69-94.
Olson, G. M., & Olson, J. S. (2000). Distance matters. *Human-computer interaction, 15*(2), 139-178.
Omoronyia, I., Ferguson, J., Roper, M., & Wood, M. (2010). A review of awareness in distributed collaborative software engineering. *Software: Practice and Experience, 40*(12), 1107-1133. doi:10.1002/spe.1005
Orlikowski, W. J. (1996). Improvising organizational transformation over time: A situated change perspective. *Information systems research, 7*(1), 63-92.
Orlikowski, W. J., & Baroudi, J. J. (1991). Studying information technology in organizations: Research approaches and assumptions. *Information systems research, 2*(1), 1-28.
Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2008, August). Distributed agile development: Using Scrum in a large project. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on* (pp. 87-95). IEEE.
Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. Proceedings of the 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.
Parviainen, P., & Tihinen, M. (2011). Knowledge-related challenges and solutions in GSD. *Expert Systems*.
Perrone, V., Zaheer, A., & McEvily, B. (2003). Free to be trusted? Organizational constraints on trust in boundary spanners. *Organization science, 14*(4), 422-439.
Petersen, K., & Wohlin, C. (2009). Context in industrial software engineering research. Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, 2009.
Philliber, S., Schwab, M., & Samsloss, G. (1980). Social Research–Guides to a Decision-making Process. Itasca, IL: Peacock: Yin.
Pichler, R. (2010). *Agile product management with scrum: Creating products that customers love*: Addison-Wesley Professional.
Pirkkalainen, H., & Pawlowski, J. (2013). Global social knowledge management: From barriers to the selection of social tools. *Electronic Journal of Knowledge Management, 11*(1), 3-17.
Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., & Beecham, S. (2012). Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology, 54*(7), 663-685. doi:10.1016/j.infsof.2012.02.006
Prause, C. R., Scholten, M., Zimmermann, A., Reiners, R., & Eisenhauer, M. (2008, August). Managing the iterative requirements process in a multi-national project using an issue tracker. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on* (pp. 151-159). IEEE.
Prikladnicki, R., Audy, J., & Evaristo, R. (2003, May). Requirements management in global software development: Preliminary findings from a case study in a sw-cmm context. In *The International Workshop on Global Software Development, ICSE* (pp. 53-58).
Prikladnicki, R., Audy, J. N. L., Damian, D., & De Oliveira, T. C. (2007, August). Distributed Software Development: Practices and challenges in different business strategies of offshoring and onshoring. In *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on* (pp. 262-274) doi:10.1109/icgse.2007.19.
Prikladnicki, R., Damian, D., & Audy, J. L. N. (2008, August). Patterns of evolution in the practice of distributed software development in wholly owned subsidiaries: A preliminary capability model. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on* (pp. 99-108). IEEE.
Prikladnicki, R., Marczak, S., Carmel, E., & Ebert, C. (2012). Technologies to Support Collaboration across Time Zones. *IEEE software*(3), 10-13.
Prinz, W., Martinez-Carreras, M., & Pallot, M. (2011). From collaborative tools to collaborative working environments. *IGI Global*, 1-10.
Prokop, L. E. (2014). A requirements-based, bottom-up SLOC estimate and analysis of NASA's Orion crew exploration vehicle spacecraft flight software. *Innovations in Systems and Software Engineering, 10*(2), 93-101.
Raatikainen, M., Männistö, T., Tommila, T., & Valkonen, J. (2011, August). Challenges of requirements engineering—A case study in nuclear energy domain. In *Requirements Engineering Conference (RE), 2011 19th IEEE International* (pp. 253-258). IEEE.

Rafii, F., & Perkins, S. (1995). Internationalizing software with concurrent engineering. *IEEE software, 12*(5), 39.

Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *Software Engineering, IEEE Transactions on*, *27*(1), 58-93. doi:10.1109/32.895989

Ravishankar, M., Pan, S. L., & Myers, M. D. (2013). Information technology offshoring in India: a postcolonial perspective. *European Journal of Information Systems, 22*(4), 387-402.

Razzak, M. A., & Mite, D. (2015, July). Knowledge Management in Globally Distributed Agile Projects--Lesson Learned. In *Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference on* (pp. 81-89). IEEE.

Regnell, B., & Brinkkemper, S. (2005). Market-driven requirements engineering for software products. In *Engineering and managing software requirements* (pp. 287-308): Springer.

Richardson, I., Avram, G., Deshpande, S., & Casey, V. (2008, August). Having a Foot on Each Shore-Bridging Global Software Development in the Case of SMEs. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on* (pp. 13-22). IEEE.

Riessman, C. K. (1993). *Narrative analysis* (Vol. 30): California. Sage Publicatoins.

RMW Report, S. E. D. (1998). *Requirements Management Guidebook* (SRM:GDBK:SWELT:1.0:14AUG98): Software Engineering Division (AIR 4.5.7).

Rodríguez, J., Vizcaíno, A., Ebert, C., & Piattini, M. (2010). Tools to support global software development processes: a survey. Proceedings of the 5th IEEE International Conference on Global Software Engineering (ICGSE), 2010

Rolland, C., Salinesi, C., & Etien, A. (2004). Eliciting gaps in requirements change. *Requirements Engineering, 9*(1), 1-15. doi:10.1007/s00766-003-0168-y

Rose, J., & Schlichter, B. R. (2013). Decoupling, re-engaging: managing trust relationships in implementation projects. *Information systems journal, 23*(1), 5-33.

Rosen, M. (1991). Coming To Terms with the Field: Understanding And Doing Organizational Ethnography*. *Journal of Management Studies, 28*(1), 1-24.

Rowland, N. J., & Gieryn, T. F. (2008). 12 Transfer Troubles: Outsourcing Information Technology in Higher Education. *Living in a Material World*, 375.

Royle, T. (2005). Realism or idealism? Corporate social responsibility and the employee stakeholder in the global fast-food industry. *Business Ethics: A European Review*.

Rubin, E., & Rubin, H. (2011). Supporting agile software development through active documentation. *Requirements Engineering, 16*(2), 117-132.

Ruhe, G. (2003). Intelligent support for selection of COTS products. In *Web, Web-Services, and Database Systems* (pp. 34-45): Springer.

Ruhe, G., & Wohlin, C. (2014). *Software Project Management in a Changing World*: Springer.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*(2), 131-164.

Rus, I., & Lindvall, M. (2002). Guest editors' introduction: Knowledge management in software engineering. *IEEE software*(3), 26-38.

Sa, L., Marczak, S., Antunes, D., & Audy, J. (2003). Quality Management as a Competitive Strategy in a Distributed Software Development Environment.

Sabaliauskaite, G., Loconsole, A., Engström, E., Unterkalmsteiner, M., Regnell, B., Runeson, P., . . . Feldt, R. (2010). Challenges in aligning requirements engineering and verification in a large-scale industrial context. In *requirements engineering: foundation for software quality* (pp. 128-142): Springer.

Salger, F., Sauer, S., Engels, G., & Baumann, A. (2010, August). Knowledge Transfer in Global Software Development-Leveraging Ontologies, Tools and Assessments. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on* (pp. 336-341). IEEE.

Sangwan, R., Bass, M., Lullick, N., Paulish, J. D., & Kazmeier, J. (2007). *Global software development handbook*. Boca Raton, FL : Auerbach Publications. Retrieved from http://marc.crcnetbase.com/isbn/9781420013856

Sarma, A., & Van Der Hoek, A. (2006, October). Towards awareness in the large. In *Global Software Engineering, 2006. ICGSE'06. International Conference on* (pp. 127-131). IEEE.

Sauer, C. (1993). *Why information systems fail: a case study approach*: Alfred Waller Ltd., Publishers.

Sawyer, P., Sommerville, I., & Viller, S. (1997). Requirements process improvement through the phased introduction of good practice. *Software Process: Improvement and Practice, 3*(1), 19-34.

Schmid, K. (2014). Challenges and Solutions in Global Requirements Engineering–A Literature Survey. In *Software Quality. Model-Based Approaches for Advanced Software and Systems Engineering* (pp. 85-99): Springer.

Schneider, S., Torkar, R., & Gorschek, T. (2013). Solutions in global software engineering: A systematic literature review. *International Journal of Information Management, 33*(1), 119-132. doi:10.1016/j.ijinfomgt.2012.06.002

Seth, F. P., Mustonen-Ollila, E., Taipale, O., & Smolander, K. (2012, December). Software quality construction: Empirical study on the role of requirements, stakeholders and resources. In *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific* (Vol. 2, pp. 17-26). IEEE.

Shah, H., Harrold, M. J., & Sinha, S. (2014). Global software testing under deadline pressure: Vendor-side experiences. *Information and Software Technology, 56*(1), 6-19.

Shaw, M. (2002, November). Self-healing: softening precision to avoid brittleness: position paper for WOSS'02: workshop on self-healing systems. In *Proceedings of the first workshop on Self-healing systems* (pp. 111-114). ACM.

Sheu, C., Chae, B., & Yang, C.-L. (2004). National differences and ERP implementation: issues and challenges. *Omega, 32*(5), 361-371. doi:10.1016/j.omega.2004.02.001

Shrivastava, S. V. (2010). Distributed agile software development: A review. *arXiv preprint arXiv:1006.1955*.

Sia, S. K., & Soh, C. (2007). An assessment of package–organisation misalignment: institutional and ontological structures. *European Journal of Information Systems, 16*(5), 568-583.

Sinha, V., Sengupta, B., & Chandra, S. (2006). Enabling collaboration in distributed requirements management. *Software, IEEE*, *23*(5), 52-61.

Šmite, D. (2006). Requirements management in distributed projects. *Journal of Universal Knowledge Management, 1*(2), 69-76.

Smite, D., Wohlin, C., Feldt, R., & Gorschek, T. (2008, August). Reporting empirical research in global software engineering: a classification scheme. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on* (pp. 173-181). IEEE.

Šmite, D., Wohlin, C., Galviņa, Z., & Prikladnicki, R. (2014). An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering, 19*(1), 105-153.

Šmite, D., Wohlin, C., Gorschek, T., & Feldt, R. (2009). Empirical evidence in global software engineering: a systematic review. *Empirical Software Engineering, 15*(1), 91-118. doi:10.1007/s10664-009-9123-y

Software-Quality-Assurance.org. (2015). *CMMi - Requirements Management (REQM)*. Retrieved 15 Jan 2015, 2015, from http://www.software-quality-assurance.org/cmmi-requirements-management.html

Soh, C., Kien, S. S., & Tay-Yap, J. (2000). Enterprise resource planning: cultural fits and misfits: is ERP a universal solution? *Communications of the ACM, 43*(4), 47-51.

Sommerville, I. (2005). Integrated requirements engineering: A tutorial. *Software, IEEE, 22*(1), 16-23.

Sommerville, I. (2011). *Software Engineering*. Boston, Massachusetts: Pearson/Addison-Wesley. Retrieved from http://books.google.co.nz/books?id=l0egcQAACAAJ

Sommerville, I., & Sawyer, P. (1997). *Requirements engineering: a good practice guide*: John Wiley & Sons, Inc.

Sparkes, A. (2005). Narrative analysis: Exploring the whats and the hows of personal stories. *Qualitative research in health care, 1*, 191-209.

Srikanth, K., & Puranam, P. (2014). The firm as a coordination system: Evidence from software services offshoring. *Organization science, 25*(4), 1253-1271.

Stake, R. E. (1995). *The art of case study research*: SAGE Publications.

Strauss, A. L., & Corbin, J. (1990). *Basics of qualitative research* (Vol. 15): Sage Newbury Park, CA.

Swan, J., Newell, S., & Robertson, M. (1999). The illusion of 'best practice'in information systems for operations management. *European Journal of Information Systems, 8*(4), 284-293.

Sydserff, R., & Weetman, P. (2002). Developments in content analysis: a transitivity index and DICTION scores. *Accounting, Auditing & Accountability Journal, 15*(4), 523-545.

Syed, J. (2008). An Islamic Perspective of Industrial Relations: The Case of Pakistan. *Journal of Management, Spirituality and Religion, 5*(4), 417-440.

Tanabe, D., Uno, K., Akemine, K., Yoshikawa, T., Kaiya, H., & Saeki, M. (2008, September). Supporting requirements change management in goal oriented analysis. In *International Requirements Engineering, 2008. RE'08. 16th IEEE* (pp. 3-12). IEEE.

Tell, P., & Babar, M. A. (2011). Requirements for an infrastructure to support activity-based computing in global software development. In *Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on* (pp. 62-69). IEEE.

Tell, P., & Babar, M. A. (2012, August). Activity theory applied to global software engineering: theoretical foundations and implications for tool builders. In *Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference on* (pp. 21-30). IEEE.

Tell, P., Babar, M. A., & Grundy, J. (2013, August). A Preliminary User Evaluation of an Infrastructure to Support Activity-Based Computing in Global Software Development (ABC4GSD). In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on* (pp. 100-109). IEEE.

Tellis, W. (1997). Application of a case study methodology. *The qualitative report, 3*(3), 1-17.

Tihinen, M., Parviainen, P., Suomalainen, T., & Eskeli, J. (2015). Study of Automated and Real-time Indicators for the Management of Global Software Development Projects. *Advances in Computer Science: International Journal*.

Torchiano, M., Jaccheri, L., Sørensen, C. F., & Wang, A. I. (2002, July). COTS products characterization. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering* (pp. 335-338). ACM.

Torkar, R., Gorschek, T., Feldt, R., Svahnberg, M., Raja, U. A., & Kamran, K. (2012). Requirements traceability: a systematic review and industry case study. *International Journal of Software Engineering and Knowledge Engineering, 22*(03), 385-433.

Turk, D., France, R., & Rumpe, B. (2014). Limitations of agile software processes. *arXiv preprint arXiv:1409.6600*.

Vaismoradi, M., Turunen, H., & Bondas, T. (2013). Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study. *Nursing & health sciences, 15*(3), 398-405.

Verner, J. M., & Abdullah, L. M. (2012). Exploratory case study research: Outsourced project failure. *Information and Software Technology, 54*(8), 866-886.

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2014). Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology, 56*(1), 54-78.

Vibha, S., Bikram, S., & Satish, C. (2006). Enabling Collaboration in Distributed Requirements Management. *Software, IEEE, 23*(5), 52-61. doi:10.1109/ms.2006.123

Von Stetten, A., Beimborn, D., Weitzel, T., & Reiss, Z. (2011). Managing the impact of differences in national culture on social capital in multinational IT project teams–a German perspective. In *Theory-Guided Modeling and Empiricism in Information Systems Research* (pp. 187-206): Springer.

Walsham, G. (1995). The emergence of interpretivism in IS research. *Information systems research, 6*(4), 376-394.

Walsham, G. (2006). Doing interpretive research. *European Journal of Information Systems, 15*(3), 320-330.

Weber, M., & Weisbrod, J. (2002). Requirements engineering in automotive development-experiences and challenges. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on* (pp. 331-340). IEEE.

Weinberg, G. M. (1995). Just say no! improving the requirements process. *American Programmer, 8*, 19-19.

Whitehead, J., Mistrík, I., Grundy, J., & van der Hoek, A. (2010). Collaborative software engineering: concepts and techniques. In *Collaborative Software Engineering* (pp. 1-30): Springer.

Wiegers, K., & Beatty, J. (2013). *Software requirements*: Pearson Education.

Wiegers, K. E. (2000). Karl Wiegers describes 10 requirements traps to avoid. *Software Testing & Quality Engineering, 2*(1).

Wiegers, K. E. (2003). Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle. *Redmond: Microsoft Press. ISBN 0-7356-1879-8*.

Winograd, T., & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*: Intellect Books.

Wolf, T., Nguyen, T., & Damian, D. (2008). Does distance still matter? *Software Process: Improvement and Practice, 13*(6), 493-510. doi:10.1002/spip.401

Yadav, V., Adya, M., Sridhar, V., & Nath, D. (2009). Flexible global software development (GSD): antecedents of success in requirements analysis. *Journal of Global Information Management*.

Yin, R. K. (2003). *Case Study Research: Design and Methods*: SAGE Publications. Retrieved from http://books.google.co.id/books?id=BWea_9ZGQMwC

Yin, R. K. (2006). Case Study Methods. In J. L. Green, G. Camilli, & P. B. Elmore (Eds.), *Handbook of complementary methods in education research*. Mahwah, NJ, US: Lawrence Erlbaum Associates.

Yin, R. K. (2014). *Case study research: Design and methods*: Sage publications.

Zahedi, M., & Babar, M. A. (2014, May). Towards an understanding of enabling process knowing in global software development: a case study. In *Proceedings of the 2014 International Conference on Software and System Process* (pp. 30-39). ACM.

Zowghi, D. (2002, May). Does global software development need a different requirements engineering process. In *Proceedings of International Workshop on Global Software Development* (Vol. 2002).

Zowghi, D., & Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and managing software requirements* (pp. 19-46): Springer.

Appendices

Appendix 1 Participant Information Sheet (Employer)

Date Information Sheet Produced: 20 September 2012

Project Title: Requirements Change Management in Global Software Development

Dear employer,

My name is Waqar Hussain and I am a PhD student working in the Software Engineering Research Lab (SERL) at Auckland University of Technology (AUT) in New Zealand. I would like to invite your organisation to take part in my doctoral research into the area of requirements management. In particular this research tries to identify and potentially address the challenges faced by global software development (GSD) practitioners.

Since this research involves understanding current practice, an essential element is partnering with expert practitioners in this area of software development. We would like to learn from your organisation. As expert practitioners in this area, you are invited to share your perspective and contribute to the body of knowledge in this area.

Please note that your participation in this research is voluntary in nature, and you may decline or withdraw your participation without any adverse consequences. None of the participants are identified nor will the information gathered be used to hamper, hinder or harm your career.

The following questions and answers are intended to address the most common questions that the participants may ask about this particular research project. If you need further information, please feel free to contact the researcher, Waqar Hussain. My contact details can be found at the end of this document. It is recommended that you use e-mail to reach me.

1. What is the purpose of this research?

Requirements management (RM) is considered as one of the most challenging areas of software development. It is one of the major factors contributing towards increased productivity, cost reduction and project success. However implementing requirements

management practices is difficult in a distributed environment where distance creates most of the challenges.

This study aims to develop a deeper understanding of requirements management practices engaged in by global software development. The purpose is to identify and (potentially rectify) challenges faced by global practitioners while performing RCM activities in globally dispersed projects. The study will also explore the helping (or hindering) role of collaborative technologies while supporting the operations of globally distributed projects.

Understanding of the challenges and current practices will suggest possible process improvements and lead to design of better support tools for collaboration. This will ultimately contribute to faster and more accurate software development.

Another purpose of this research is to contribute to the completion of my PhD degree. We are intending to present the findings from this research in academic conferences and journal papers. In doing so, we will aggregate the findings so that your observations or comments will not be linked to you personally. You have the option to verify the accuracy of any transcripts made. Neither the organization nor the participants will be identified.

2. How was I identified and why am I being invited to participate in this research?

Your organization has been invited to participate in this research because of its expertise in the area of requirements management. With your assistance, at this stage, we would like to recruit software practitioners from your organization based on their overall experience in software development. We recommend that a participant would have a 5 year or more over all experience in software development with at least one year experience of working on a global software development project. Based on my knowledge of the organization we anticipate approximately four to six participants from the management, development and quality assurance area of the development project.

3. What will happen in this research?

Your organization will be asked for a written consent to participate in the research. You can do this by signing this Participant Information Sheet. Once I have received your approval, you would forward individual research invitation to the suitable participants within your organization for taking part in this study. The interested participants will

contact me directly via email (which is wholly voluntary on their part) to join the research and avail the opportunity of being part of a global research project aiming to improve the state of GSD.

- Data Collection

For this study several different sources of information will be used to gather data. Data collection will be done for the project(s) whose formal requirements specification is already base-lined. Data will be collected from various artefacts (both electronic and non-electronic), participant interviews, discussions and will be collectively used for analysis. In some cases, I may need to contact your organization or research participants through phone calls or by face to face meetings.

- Interviews

For each participant we envisage an interview session (on two visits around four month apart) of one hour followed up by a further session, after a group of interviews have been completed, to confirm the understandings and check the transcripts (it is estimated that this follow up session will be less than one hour in duration).

- Analysis

The data collection will be followed by an analysis phase from which we hope to develop insights into challenges faced by the practitioners, the role of collaborative technology and the practices which contribute to effective requirements change management in global software development.

4. What are the discomforts and risks?

Based on the prior knowledge of the organization and the highly focused nature of the research I have a view that the potential discomforts and risks are minimal. The organization may be worried about commercial sensitivity and getting identified through the research work. You may feel uncomfortable that my presence of may inhibit others.

5. How will these discomforts and risks be alleviated?

I will be vigilant to the fact that the anonymity of the participants as well as the organization is maintained at all stages of the research. The organization and participants will not be identified or referred to by their names in the research. I will interact with the participants for data collection (which is mainly done through electronic resources) in as non-disruptive as possible manner to work-as-usual.

In order to manage disclosures of commercially sensitive information a draft or transcript will be provided to you for review prior to the publication. The procedure described below in the section 'how will my privacy be protected?' will be employed to further guard participants from any potential risk or discomfort.

6. What are the benefits?

The insights gained from this research will be available for your organization, the participating individuals and their colleagues. It will also add to the existing body of knowledge of current requirements management issues and influencing practices in global software development area. The 'issue diagnoses' performed by this research, may act as a rectification base to address the current and future collaboration issues of an organization.

It is expected that research will help in self-assessment for the organizations and the participants and aid in the adoption of effective practices. This can enable them to realize and leverage potential benefits GSD offers; cutting down development costs, and improving their product's time to market. The participating organizations can align their existing processes with a guiding set of good practices (provided by this research) that is theoretically informed, practically validated, based on expert advice and suited to GSD work.

With the newly gained knowledge about suitable technologies to manage requirements over distance, the participants can better envisage and orchestrate collaborative activities in a distributed environment. It is expected that the outcome of this research should also contribute benefits to the ICT sector in general, through generating knowledge specific to local context. In addition I will also benefit from this research, since it will contribute to the completion of my PhD degree.

7. How will my privacy be protected?

Privacy and confidentiality will be respected and protected with diligence and by all available means. The identity of participants will be protected at all stages of a project to the extent possible. This will be done by coding of data and removal of identifying material from documentation. Furthermore the participants will be addressed by their group roles rather than individual roles to de-identify them and maintain their anonymity. The participant will be provided with the transcripts for their confirmation to avoid any conflict of interest and make sure the transcript does not contain any content which might reveal their identity.

The participants or the organization will not be identified by name; rather a hypothetical name will be used whenever a referral is made to the organization or the participants in subsequent publications (report, thesis, journal or articles etc.).

8. What are the costs of participating in this research?

The research will provide the opportunity for the participating organization to benefit from its results without a major cost (in time and resources) and to have an in depth investigation of a critical area of GSD practice with the potential to contribute to future development effectiveness.

9. What opportunity do I have to consider this invitation?

We would appreciate a response within a week.

10. How do I agree to participate in this research?

Your participation and feedback will be of great value in helping us identify more effective ways to manage requirements and develop computer software in global distributed software development environment. If you are willing to participate in this research, please return the enclosed consent form within one week of receiving this invitation.

11. Will I receive feedback on the results of this research?

The research findings will be disseminated via the standard academic channels, published PhD thesis, related seminars, conference and journal papers. All willing participants will receive a summary of the research findings either as an extract of the thesis or as developed in academic publications.

12. What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Project Supervisor, Tony Clear, tclear@aut.ac.nz, (0064) 921 9999 ext 5329

Concerns regarding the conduct of the research should be notified to the Executive Secretary, AUTEC, Dr Rosemary Godbold, rosemary.godbold@aut.ac.nz , (0064) 921 9999 ext. 6902.

13. Whom do I contact for further information about this research?

Researcher Contact Details: Waqar Hussain

Software Engineering Research Laboratory (SERL)

Phone: +64 9 921 9999 Ext 5852

Email: waqar.hussain@aut.ac.nz

Project Supervisor Contact Details:

Dr Tony Clear

Associate Professor Associate Dean Research

Faculty of Design & Creative Technologies AUT University

Phone : 64 9 921 9999 Ext 5329

Email address: tony.clear@aut.ac.nz

Approved by the Auckland University of Technology Ethics Committee on type the date final ethics approval was granted, AUTEC 12/287

Appendix 2 Participant Information Sheet (Individual)

Date Information Sheet Produced: 20 September 2012

Project Title: Requirements Change Management in Global Software Development

Dear potential participant,

My name is Waqar Hussain and I am a PhD student working in the Software Engineering Research Lab (SERL) at Auckland University of Technology (AUT) in New Zealand. I would like to invite your organisation to take part in my doctoral research into the area of requirements management. In particular this research tries to identify and potentially address the challenges faced by global software development (GSD) practitioners.

Since this research involves understanding current practice, an essential element is partnering with expert practitioners in this area of software development. As practitioners in this area, you are invited to share your perspective and contribute to the body of knowledge in this area.

Please note that your participation in this research is voluntary in nature, and you may decline or withdraw your participation without any adverse consequences. You will not be identified nor will the information gathered be used to hamper, hinder or harm your career.

The following questions and answers are intended to address the most common questions that you may have about this particular research project. If you need further information, please feel free to contact the researcher, Waqar Hussain. My contact details can be found at the end of this document. It is recommended that you use e-mail to reach me.

1. What is the purpose of this research?

Requirements management (RM) is considered as one of the most challenging areas of software development. It is one of the major factors contributing towards increased productivity, cost reduction and project success. However implementing requirements management practices is difficult in a distributed environment where distance creates most of the challenges.

This study aims to develop a deeper understanding of requirements management practices engaged in by global software development The purpose is to identify and (potentially rectify) challenges faced by global practitioners while performing RCM activities in globally dispersed projects. The study will also explore the helping (or hindering) role of collaborative technologies while supporting the operations of globally distributed projects.

Understanding of the challenges and current practices will suggest possible process improvements and lead to design of better support tools for collaboration. This will ultimately contribute to faster and more accurate software development.

Another purpose of this research is to contribute to the completion of my PhD degree. I intend to present the findings from this research in academic conferences and journal papers. In doing so, the findings will be aggregated so that your observations or comments will not be linked to you personally. You have the option to verify the accuracy of any transcripts made. Neither the organization nor any member of your organization participating in the research will be identified

2. How was I identified and why am I being invited to participate in this research?

Your organization has been invited to participate in this research because of its expertise in the area of requirements management.

You are invited to participate in this research because you are (or have been) part of a team working in global software development project and have adequate work experience. Since you have accepted on open invitation to participate in this research, your participation will remain anonymised to your employer. Participation in this research is voluntary. Your selection is based on your prior experience in global projects.

3. What will happen in this research?

You are invited to take part in this research by sharing your experiences while working in globally distributed projects. You will take part in an interview to discuss your role and responsibilities working in a distributed environment.

I envisage an interview session (on two visits around six month apart) of one hour with you followed up by a further session, after a group of interviews have been completed, to

confirm the understandings and check the transcripts (it is estimated that this follow up session will be less than one hour in duration). The information obtained from these sessions will be anonymised. You will also have the opportunity to review the transcripts or draft before the research results are published, for confirmation to make sure they don't contain any content which might reveal your identity.

The data collection through various means (artefacts, interviews and observations) would be followed up by an analysis phase from which we hope to develop insights into challenges faced by the practitioners, the role of collaborative technology and the practices which contribute to effective requirements change management in global software development.

4. What are the discomforts and risks?

Based on the prior knowledge of the organization and the highly focused practice nature of the research It is believed that the potential discomforts and risks are minimal. However;

- You may feel uncomfortable being that your employer will know who is participating in the study and who has elected not to take up the invitation
- You may feel uncomfortable having notes taken.

5. How will these discomforts and risks be alleviated?

In order to alleviate the first area of discomfort you will be reminded of our assurance of voluntary nature of participation to you at regular intervals throughout the fieldwork.

The second possible area of discomfort will be addressed by emphasising that the data being collected to notes taken relate to information related to the current practices in general rather than an individual's activity. Personal performance will not be judged.

I will be vigilant to the fact that your anonymity is maintained at all stages of the research and you do not experience any discomfort or pressure of any type. The procedure described below in the section 'how privacy will be protected' will be employed to further guard you from any potential risk or discomfort.

6. What are the benefits?

The ‘issue diagnoses’ performed by this research, may act as a rectification base to address the current and future collaboration issues of an organization.

It is expected that research will help in self-assessment for you and your organization to aid in the adoption of effective software development and requirements management practices. This can enable you to realize and leverage potential benefits GSD offers; cutting down development costs, and improving their product’s time to market.

Your organization can align its existing processes with a guiding set of good practices (provided by this research) that is theoretically informed, practically validated, based on expert advice and suited to GSD work.

With the newly gained knowledge about suitable technologies to manage requirements over distance, you can better envisage and orchestrate collaborative activities in a distributed environment.

It is expected that the outcome of this research should also contribute benefits to the ICT sector in general, through generating knowledge specific to local context. In addition I will also benefit from this research, since it will contribute to the completion of my PhD degree.

The insights gained from this research will be available for you and your organization. It will also add to the existing body of knowledge of current requirements management issues and influencing practices in global software development area.

7. How will my privacy be protected?

Your privacy, identity and confidentiality will be respected and protected with diligence and by all available means and at all stages of the research to the extent possible. This will be done by coding of data and removal of identifying material from documentation. Furthermore, you and your colleagues will be addressed by their group roles rather than individual roles to de-identify them and maintain their anonymity. You will be provided with the transcripts for their confirmation to avoid any conflict of interest and make sure the transcript does not contain any content which might reveal their identity.

You will not be identified by name; rather a hypothetical name will be used whenever a referral is made to you in subsequent publications (report, thesis, journal or articles etc.).

8. What are the costs of participating in this research?

The research will provide the opportunity for you to benefit from its results without a major cost (in terms of time and resources) and to have an in depth investigation of a critical area of GSD practice with the potential to contribute to future development effectiveness.

- What opportunity do I have to consider this invitation?
- We would appreciate a response within 48 hours of receiving this invitation.
- How do I agree to participate in this research?

Your participation and feedback will be of great value in helping us identify more effective ways to manage requirements and develop computer software in global distributed software development environment. If you are willing to participate in this research, please return the enclosed consent form within one week of receiving this invitation.

9. Will I receive feedback on the results of this research?

The research findings will be disseminated via the standard academic channels, published PhD thesis, related seminars, conference and journal papers.

You will receive a summary of the research findings either as an extract of the thesis or as developed in academic publications, if desired.

10. What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Project Supervisor, Tony Clear, tclear@aut.ac.nz, (0064) 921 9999 ext 5329

11. Concerns regarding the conduct of the research should be notified to?

The Executive Secretary, AUTEC, Dr Rosemary Godbold, rosemary.godbold@aut.ac.nz , (0064) 921 9999 ext. 6902.

12. Whom do I contact for further information about this research?

The Researcher.
Researcher Contact Details:

Waqar Hussain

Software Engineering Research Laboratory (SERL)

Phone: +64 9 921 9999 Ext 5852
Email: waqar.hussain@aut.ac.nz

Project Supervisor Contact Details:

Dr Tony Clear

Associate Professor Associate Dean Research

Faculty of Design & Creative Technologies AUT University

Phone : 64 9 921 9999 Ext 5329

Email address: tony.clear@aut.ac.nz

Approved by the Auckland University of Technology Ethics Committee on 11 January 2013 AUTEC Reference, AUTEC Reference number 12/287

Appendix 3 Consent Form

For use when interviews are involved.

Project title: Requirements Change Management in Global Software Development

Project Supervisor: Dr. Tony Clear

Researcher: Waqar Hussain

I have read and understood the information provided about this research project in the Information Sheet dated 26 September 2012.

○ I have had an opportunity to ask questions and to have them answered.

○ I understand that contributions made in interviews and observations augmented by development artefacts, associated electronic communications and research diary notes taken by the researcher, will be analyzed in order to better understand the collaboration process along with the related practices.

○ I allow the interviews to be audio-taped Yes ○ No ○

○ I understand that I may withdraw myself or any information that I have provided for this project at any time prior to completion of data collection, without being disadvantaged in any way.

○ If I withdraw, I understand that all relevant information including tapes and transcripts, or parts thereof, will be destroyed.

○ I agree to take part in this research.

○ I wish to receive a copy of the report from the research (please tick one): Yes○ No○

Participant's signature:..... ……

Participant's name: …….....……

Participant's Contact Details (if appropriate):

……..

……..

Date :

Approved by the Auckland University of Technology Ethics Committee on 11 January 2013 AUTEC Reference number 12/287

Appendix 4 Interview Schedule

Date of Interview Schedule Produced: 26 September 2012 (Revised on the basis of slight modifications made to the original schedule during interviews)

Project Title: Requirements Change Management in Global Software Development

This study aims to develop a deeper understanding of the Requirements Engineering and Requirements Management practices engaged in by global software development practitioners and the helping (or hindering) role of collaborative technologies. The purpose is to identify and (potentially address) challenges faced by global practitioners while performing RCM activities in globally dispersed projects.

Introduction

This proposed interview schedule is indicative only, the aim is to gather participants understanding of the problem domain for a specific software development project, project context, or clarifying an incident that occurred.

Ice Breaker: Explaining researcher's role, ethics approval process and confidentiality of information collected.

Personal Information: Job Title

Project Related: Project Involved, Role and Responsibilities, Duration of Involvement, Experience

Question 1: What would you say is the primary purpose of requirements change management?

Question 2: In a few words how would you summarize the general aim of managing requirements change?

Question 3: What are the major challenges in managing requirements in a globally distributed development environment?

Question 4: Can you explain the process through which a change goes through from when it is initiated to when it is deployed into the production?

Question 5: In general, how did distance (geographical, temporal, socio-cultural) impact the management of these activities?

Question 6: What is the most important problem in requirements change management and requires urgent improvement?

Question 7: What combination of collaborative technologies do you use for requirements management?

Question 8: Are you satisfied with the technology role or you have some suggestions for improvement?

Question 9: If you are not satisfied, what do you suggest should be done to address it?

Question 10: Think of an incident (both positive or negative) you experienced that was significant in terms of its positive or negative impact on project schedule or deadline?

Question 10-a: What were the general circumstances leading up to this incident?

Question 10-b: What exactly was done by a team member that was so helpful at that time?

Question 10-c: Think of the role collaborative technology played in the incident and answer how did:

10-c-1: The technology allowed to collect all the information needed to carry out a specific task?

10-c-2: The technology allowed suspending and resuming activities to perform alternative tasks?

10-c-3: The technology allowed resuming an activity on different workstations and/or devices connected to the system?

11-c-4: The technology allowed sharing of activities with other members?

Question 10-d: Why was this so helpful in getting your team's job done?

Question 10-e: When did this incident happen?

Appendix 5 Interview Details

Formal Interviews					
	Participant Pseudonym	*First Round Interviews*	*Second Round Interviews*	*Venue*	*Site*
Case 1	WAA	1 hr 0 min 48 sec	32 min 53 sec	Company Meeting Room	Pakistan
	MAI	1 hr 06 min 52 sec	X	Company Meeting Room	Pakistan
	IAE	0 hr 40 mins 0 sec	X	CEO Office	Pakistan
	ZAE	1 hr 09 min 22 sec	1 hr 07 min 19 sec	Company Meeting Room	Pakistan
	OOI	0 hr 52 mins 0 sec	52 min 07 sec	Company Meeting Room	Pakistan
	IME	1 hr 09 min 27 sec	51 min 16 sec	Company Meeting Room	Pakistan
	DUI	1 hr 58 min 57 sec	55 min 35 sec	Company Meeting Room	Pakistan
	DUA	x	1 hr 10 min 0 sec	Skype	Pakistan
		7 hr 57 min 26 sec	**4 hr 19 min 10 sec**		
Case 2	ICK	1 hr 01min 30 sec	0 hr 35 min 32 sec	University meeting Room	New Zealand
	LEC	1 hr 06 min 58 sec	0 hr 31 min 32 sec	Interviewees University Office	New Zealand
	CKE	0 hr 52 min 51 sec	0 hr 31 min 55 sec	University meeting Room	New Zealand
	MOL	1 hr 07 min 38 sec	0 hr 55 min 04 sec	University meeting Room	New Zealand
	ILE	1 hr 02 min 14 sec	0 hr 45 min 49 sec	Interviewees University Office	New Zealand
	IRW	1 hr 01 min 43 sec	0 hr 42 min 43 sec	Interviewees University Office	New Zealand
	ISE	1 hr 01 min 03 sec	X	University meeting Room	New Zealand
	AVA	1 hr 23 min 34 sec	X	Skype Meeting	USA
	IZN	0 hr 40min 35 sec	X	University meeting Room	New Zealand
		9 hr 18 min 6 sec	**4 hr 2 min 35 sec**		

Informal Interviews					
Case 1	DUI	22.04 mins	X	Outdoor/Restaurant	Pakistan
	DUI	X	08 min 55 sec	Company Meeting Room	Pakistan
	DUI	X	23 min 03 sec	Company Meeting Room	Pakistan
	Team	X	51 min 26 sec	Outdoor/Restaurant	Pakistan
Case 2	MOL	X	1hr :30 mins	Client Organization	New Zealand
		0 hr 22 min 04 sec	**2 hr 53 min 24 sec**		

Total	33 Interviews

,

Appendix 6 Participant Demographic Information

Demographic Information of Participants				
	Role	*Gender*	*Age (appx)*	*Location*
Case 1	Developer	Male	32	Pakistan
	Developer	Male	29	Pakistan
	Business Analyst	Male	50	USA
	Team Lead	Male	33	Pakistan
	Project Manager	Male	38	Pakistan
	Business Analyst	Female	34	Pakistan
	Development Manager	Male	35	Pakistan
Case 2	Project Manager	Male	50	New Zealand
	Product Owner	Female	55	New Zealand
	Business Analyst/ Tester	Female	38	New Zealand
	Business Analyst/ Tester	Male	39	New Zealand
	Research Director	Female	55	New Zealand
	Internal Auditor	Male	60	New Zealand
	Business Analyst/ Document Engineer	Male	39	New Zealand
	Business Analyst	Male	50	USA
	IT Director	Female	48	New Zealand

Appendix 7 Ethics Approval

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

A U T E C
S E C R E T A R I A T

16 January 2013

Tony Clear
Faculty of Design and Creative Technologies

Dear Tony

Re Ethics Application: **12/287 Requirements change management in global software development.**

Thank you for providing evidence as requested, which satisfies the points raised by the AUT University Ethics Committee (AUTEC). Please forward evidence of the organisations agreement once received.

Your ethics application has been approved for three years until 11 January 2016.

As part of the ethics approval process, you are required to submit the following to AUTEC:

- A brief annual progress report using form EA2, which is available online throughhttp://www.aut.ac.nz/research/research-ethics/ethics. When necessary this form may also be used to request an extension of the approval at least one month prior to its expiry on 11 January 2016;
- A brief report on the status of the project using form EA3, which is available online throughhttp://www.aut.ac.nz/research/research-ethics/ethics. This report is to be submitted either when the approval expires on 11 January 2016 or on completion of the project.

It is a condition of approval that AUTEC is notified of any adverse events or if the research does not commence. AUTEC approval needs to be sought for any alteration to the research, including any alteration of or addition to any documents that are provided to participants. You are responsible for ensuring that research undertaken under this approval occurs within the parameters outlined in the approved application.

AUTEC grants ethical approval only. If you require management approval from an institution or organisation for your research, then you will need to obtain this. If your research is undertaken within a jurisdiction outside New Zealand, you will need to make the arrangements necessary to meet the legal and ethical requirements that apply there.

To enable us to provide you with efficient service, please use the application number and study title in all correspondence with us. If you have any enquiries about this application, or anything else, please do contact us at ethics@aut.ac.nz.

All the very best with your research,

Dr Rosemary Godbold
Executive Secretary
Auckland University of Technology Ethics Committee

Cc: Waqar Hussain waqar.hussain@aut.ac.nz