UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
Electronics and Computer Science

Investigating Software Customisation across Distributed Boundaries

by

Abdulrahman M. Qahtani

Thesis for the degree of Doctor of Philosophy in Computer Science

February 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Doctor of Philosophy

Investigating Software Customisation across Distributed Boundaries

Abdulrahman M. Qahtani

In recent years the software industry has paid significant attention to customising software products by means of outsourcing and Agile development practices. When these areas overlap there are benefits and challenges. This study investigates the customisation process for packaged software products in projects involving multiple clients, and the communication of their requirements across distributed boundaries. A literature review identified the challenges involved and a framework for challenges of customising software products across distributed boundaries (FCCSD) is proposed to address them using onsite development practices. Local decision making and local development are considered as a means of reducing difficulties in communicating the customisation requirements of multiple clients across distributed boundaries through a new model. This model for communicating customisation requirements in the distributed domain, termed (CCRD), has two scenarios, one using decision making at the distributed client's location, and the second enhancing onsite development of certain requirements in order to reduce delays and misunderstandings between the clients and the teams involved.

A preliminary study was conducted to confirm the FCCSD. This employed a questionnaire survey of 19 highly experienced participants. The CCRD model was evaluated in three stages using an industrial case study of a company with 18 distributed clients. The first was a contextual inquiry to create a baseline model for a real world case. The second involved the simulation of the CCRD model using a discrete-event simulation approach, traced by the baseline model using real data. Finally, the findings of previous experiments were validated through a study conducted by means of semi-structured interviews with seven experts at this same company.

The key contributions of this study are as follows: First is the proposed framework (FCCSD) that addresses a number of challenges facing customisation across distributed boundaries from literature. Second, this study highlights issues of the communication and negotiation of clients' customisation requirements across distributed boundaries, and their implications.

Therefore, this study demonstrates the statistically significant impact of making decisions and negotiating clients' requirements, as well as conducting certain development practices, on their premises to limit the implications of communication challenges over distributed boundaries, such as delays in making decisions, a long duration of development and the entire customisation process, and misunderstandings about clients' requirements.

Table of contents

List of tables

List of figures

Declaration of authorship

I, Abdulrahman Qahtani, declare that this thesis entitled 'Investigating Software Customisation across Distributed Boundaries' and the work presented in it are my own and have been generated by me as the result of my own original research. I confirm that:

1. This work was done wholly or mainly while a candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:

- Qahtani, A. M., Gravell, A. M., & Wills, G.B., 2012 Toward a framework for software development across organizational boundaries, in *Proceedings of the XP 2012 Doctoral Symposium*, 13th International Conference on Agile Software Development, Malmö.
- Qahtani, A.M., Wills, G. B. & Gravell, A. M., 2012 Toward a framework for the development and specialisation of product software across organisational boundaries, in *International Conference on Information Society* (i-Society, 2012), IEEE, London, pp. 391–395.
- Qahtani, A. M., Wills, G. B. & Gravell, A. M., 2013 A framework of challenges and key factors for applying Agile methods for the development and customisation of software products in distributed projects. *International Journal of Digital Society (IJDS)*, Special Issue, 1(1), ISSN 20402570.
- Qahtani, A. M., Wills, G. B. & Gravell, A. M., 2013 Customisation software products in distributed software development: A model for allocating customisation requiremnts across organisationsal boundaries, in *International Conference on Information Society* (i-Society 2013), IEEE, Toronto, pp. 95–101.
- Qahtani, A. M., Wills, G. B. & Gravell, A. M., 2015 The impacts of local decision making on customisation process speed across distributed boundaries: A case study. *International Journal of Computer, Information, System and Control Engineering,* 9(1), 2015.

Signed: .....

Date:.....

Acknowledgements

I would like to express my deep gratitude to my supervisors, Gary Wills and Andy Gravell, for taking me as a research student under their valuable supervision. Their advice, discussions and guidance were the real encouragement to complete this work. I admire their simplicity, generosity and work ethics.

I wish also to extend my sincere gratitude to the organisations and individuals that have been involved in and cooperated with this study, answering questionnaires and participating in this study.

I would like to thank my family: my parents for their endless support and praying for me, my brothers and sisters for all their support and encouragement. I would also like to thank my wife and my children for understanding, supporting me through all stages of my research and also for travelling with me far away from home and for putting a smile on my face, even in difficult times.

Finally, I would like to acknowledge the support of the Kingdom of Saudi Arabia through Taif University for this scholarship to enable me to pursue the PhD study.

Definitions and abbreviations

CCRD Communicating the Customisation Requirements of multi-clients in a Distributed domain

COTS Commercial off-the-shelf Software

CRM Customer Relationship Management

DAD Distributed Agile Development

DSD Distributed Software Development

ERP Enterprise Resource Planning

FCCSD Framework for Challenges of Customising Software Products across Organisational Boundaries in DSD

GSD Global Software Development

IT Information Technology

PC Personal Computer

PS Packaged Software

RE Requirement Engineering

XP Extreme Programming

Chapter 1: Introduction

The software industry has turned to performing software engineering around the world, and tends to decentralise and distribute both development and maintenance in a phenomenon called global software development (GSD) (Sengupta, Chandra & Sinha, 2006). Distributed software development (DSD), including GSD, has received much attention owing to the importance of increasing distributed development and outsourcing projects (Schneider et al., 2012; Jiménez et al., 2009). The main motive for this trend is the need to allocate development teams to different distributed locations in order to achieve higher quality products and services at lower cost (Khan, Niazi & Ahmad, 2009), looking for competitive advantage in terms of quality, cost and human resources (Ranganathan & Balaji, 2007). The transition from co-located to distributed projects has been accompanied by significant changes to aspects of the software development field (da Silva, França & Prikladnicki, 2010), including features and challenges integral to software development project practices (Jiménez et al., 2009).

In order to meet the challenges associated with global adoption, changes are needed in software engineering practice. Requirements engineering (RE) plays an important role in the software development process in both the collocated and the distributed domains (Damian & Zowghi, 2003). Managing the communication of client requirements is a key component of the development production process in the marketplace. While it is difficult to negotiate and communicate these requirements locally, it is even more difficult to do so over distributed boundaries, particularly for multiple clients. This difficulty increases in DSD projects as well as projects that have multiple distributed clients across organisational and distributed boundaries. In the last two decades there has been a significant transition from forms of collocated development to GSD, requiring more communication across organisational boundaries (Damian, 2007). RE in a distributed domain is a complex intersection phenomenon that encompasses numerous technical, social and organisational aspects (Damian, 2002). Since 2000 a number of phenomena and concepts have been growing significantly in the software industry and market that have affected several aspects of software engineering, such as Agile software development methods (Erickson, Lyytinen & Siau, 2005), an increase in DSD projects (Prikladnicki & Audy, 2010), and growth in using and producing packaged software (PS) products (Morabito, Pace & Previtali, 2005). Consequently, research has focused on the impact of these phenomena on different activities in the software development life cycle in order to identify and discuss the opportunities and challenges of this change.

1.1 Motivation for the Research

The software industry provides different types of products and processes (Xu & Brinkkemper, 2007). Some products are built from scratch and based on clients' requirements, such as bespoke software products, whereas others are sold 'off-the-shelf' then customised, such as PS products. Much research has indicated the importance of the customisation process for PS products in the industry, especially in recent years with the significant growth in demand for this type of software (Gregory & Baskerville, 2014) (section 2.3.1). Numerous aspects are discussed and explored in the literature, such as distributed development and RE, especially RE of bespoke software products compared to PS products (see section 2.3). Recent research indicates a lack of studies that focus on RE practices of PS products, including the processes and tools of this domain (Jebreen, 2014) (section 2.3.1).

The intersection between the customisation of software products, distributed development and Agile development domains sketched in order to discuss the RE challenges and difficulties has identified an interesting area scarcely represented in the research literature. Many studies discuss the challenges and opportunities of RE with DSD, or RE with Agile with DSD projects (Chapter 2), however the process of customising software products in the distributed domain using a lighter development approach such as the Agile development approach has not been investigated as much as bespoke software products (Jebreen & Wellington, 2013), as described in section 2.3.1. Therefore, the scope of this thesis lies in the customisation of software products in the distributed domain by identifying the challenges and key factors in the software customisation process across organisational and distributed boundaries, using local development practices such as the Agile approach. It investigates the impact of making decisions and undertaking development practices at clients' locations to reduce the challenges of communicating multiple clients' customisation requirements over distributed boundaries.

1.2 Research Questions and Hypotheses

The research questions and hypothesis defined for this research are:

RQ1: What are the challenges identified in the literature in customising software products across organisational boundaries in the distributed domain?

The customisation of software products faces difficulties in projects involving multiple clients across the organisational and distributed domains. Thus, this research question aims to investigate the challenges and difficulties identified in the literature on customising software products across these boundaries. Chapter 3 presents and confirms a Framework for Challenges

of Customising Software Products across Organisational Boundaries in DSD (FCCSD) that synthesises the findings in the literature on RE, DSD and the Agile development approach to the process, answering this research question by addressing the key challenges and factors to help researchers and practitioners understand and consider these challenges. In addition, Chapter 7 discusses the FCCSD in detail.

RQ2: What is the impact of making decisions locally on the total decision period and the total customisation period of software customisation processes in the distributed domain?

Communicating customisation requirements for multiple clients involves issues such as holdups due to misunderstandings over clients' requirements across distributed boundaries, impacting on the customisation process. Therefore, this research question aims to investigate the impact of making decisions onsite on the duration of both decision making and the entire customisation process for the distributed multiple-client domain. This question has been answered by confirming the two defined hypotheses.

Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients.

Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients.

Allocating negotiation and decision making to the client's site minimises the amount of further communication necessary between clients and development teams. Therefore, Hypothesis 1 is concerned with the impact of adopting local decision making on the duration of decision making when multiple clients are involved in a customisation project. Chapter 4 presents a model, Communicating Customisation Requirements of Multi-Clients in the Distributed Domain (CCRD), to reduce the challenges. In addition, section 6.2.3 presents the evaluation that confirms Hypothesis 1, as discussed in section 7.4.1. Hypothesis 2 indicates the impact of local decision making on the entire customisation process by reducing its duration. In Chapter 6, section 6.2.4 presents the evaluation experiment that confirms Hypothesis 2, as discussed in section 7.4.2.

RQ3: How much would undertaking the process of development of minor tasks at clients' locations reduce the total development and customisation period?

Undertaking the development of minor customisation requirements at a client's location plays a vital role in shortening the development process in customisation projects. This research question aims to examine the impact of doing so on the length of the development period and

the entire duration of customisation in the distributed multi-client domain. This research has been answered by confirming the hypothesis defined for its solution.

Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process.

Hypothesis 3 is concerned with the impact of allocating development of minor customisation requirements to clients' locations to cut the duration of both development and the entire customisation process. Section 6.3 presents the results of the evaluation for local development process at clients' locations, which confirmed this hypothesis as discussed in section 7.4.

1.3 Contributions

This thesis investigates customising software products across organisational and distributed boundaries to discuss and provide a solution to the challenge of communicating the requirements of multiple clients across the distributed domain. Through the methodology used in this research and its findings, the key contributions are as follows.

- It has identified in the literature the challenges and factors of customising software products across organisational and distributed boundaries and presents them in the form of a confirmed framework, the FCCSD (Figure 3.1). This informs researchers and practitioners of the key issues for customisation projects, especially those adopting local practices as in the Agile development approach, with multiple clients in the distributed domain. The FCCSD is discussed in section 7.2.

- It investigates the impact of adopting local decision making at the clients' location on the customisation process, by designing the CCRD model (Figure 4.1) for communicating customisation requirements for multiple clients over the distributed domain. This contribution is discussed in section 7.4.

- It extends the CCRD model by allocating the development process to the clients' location (Figure 4.2). It then investigates the impact using simulation to develop certain requirements for the duration of the development process and the entire customisation process in order to reduce the challenges of communicating clients' requirements over distributed boundaries. This contribution is discussed in section 7.5.

- The simulation of the baseline model represents a practical contribution of this research, which models an industrial case study for customising software for multiple clients in

the distributed domain with a central development team, using the real requirements of 18 clients in 2479 requirements over 1290 working hours.

- Another simulation model is provided for the customisation process using local decision making and a local development process using real client data from the selected case study, which in future work is to be extended and used in other research projects, as suggested in section 8.4.

- In doing so the three research questions above are answered and give statistically significant support for all three hypotheses.

1.4 Thesis Structure

This thesis is divided into eight chapters. After Chapter 1, the thesis is organised as follow.

Chapter 2 **Review of customisation software products across distributed organisational boundaries:** This chapter presents the literature in three areas involved in the development process: DSD; RE; and Agile software development. In addition, it presents the overlap of these areas in order to investigate the effect of local negotiation and decision making on the customisation requirements of distributed development projects. It concludes with a table displaying the challenges and factors impacting on DSD customisation, as identified in the literature.

Chapter 3 **Framework for challenges of customising software products across organisational boundaries in DSD:** This chapter presents the FCCSD, addressing the identified challenges of customising software products across organisational boundaries in distributed development projects. It presents the process design, then discusses its components in detail. A preliminary confirmatory study was conducted in the form of a questionnaire with 19 researchers and practitioners from the software development community in order to confirm the importance of the process of customising software products as a domain, and also its challenges, presented though the FCCSD.

Chapter 4 **A model for communicating customisation requirements of multi-clients in distributed domain:** This chapter describes the importance of communication of customisation requirements for multiple clients across distributed boundaries. Also, it provides a model for the customisation process in the distributed domain. This CCRD model has two scenarios, one using local decision making at the distributed clients' location, and the second enhancing the application of the development process for some clients' requirements at their location.

Chapter 5 **Evaluating local decision making and development at client's location in DSD:** This chapter presents the methods used in this research in order to build the environment for the evaluation process and to complete it for the CCRD model, to examine the defined hypothesises for this research. This includes a sequence of activities starting with selecting a case study, conducting a contextual review study, building a baseline model for the selected case study, validating the CCRD model, tracing the baseline model and finally confirming the findings of the experiments in a confirmatory study.

Chapter 6 **Evaluation experiments:** This chapter presents the evaluation of this research, including the settings used, the design of evaluation experiments and the results obtained by each experiment. It presents findings and interpretation of three experiments: evaluation for CCRD model – local decision making; evaluation for CCRD model – local development; and findings of the confirmatory study.

Chapter 7 **Discussion**: This chapter discusses the research findings, starting from the identified challenges of the software customisation process for projects undertaken across organisational and distributed boundaries through the FCCSD and the preliminary confirmatory study conducted to confirm the framework's components. It also presents a discussion of the evaluation results for the CCRD model, both for local decision making and local development, supported by the findings of the confirmatory study that use the results to answer the research questions and hypotheses of this research and considering the main research limitations.

Chapter 8 **Conclusion and future work**: This chapter presents the objectives of this research and an overview of the research activities. It goes on to present the research findings based on the research questions and defined hypotheses. In addition, it makes some suggestions to extend this research in future in terms of new ideas and improved simulation models for the customisation process in the distributed domain. Finally, this chapter concludes with the key outcomes of this research.

Chapter 2: Review of Customisation Software Products across Distributed Organisational Boundaries

2.1 Introduction

A literature review has three important functions: to provide the background to the study for the researcher; to present important research in the relevant area; and to establish the study's position by linking it to a chain of research and expanding knowledge in a particular research area (Weissberg & Buker, 1990). This chapter presents a literature review of various topics relating to this research. Its main goal is to review the customisation of software products across distributed boundaries in a DSD environment by reviewing the research in related disciplines.

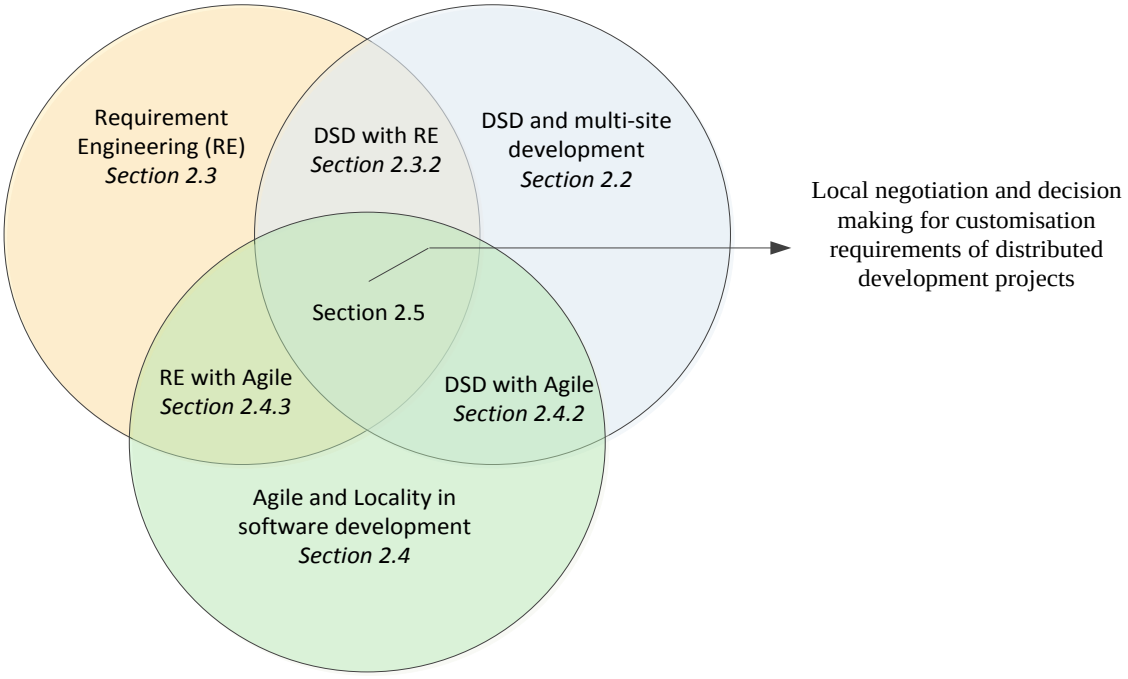


Figure 2.1: Software development areas reviewed

Figure 2.1 illustrates the three areas reviewed in this chapter. First is DSD, incorporating research discussing the shift of development projects from collocated to distributed, including research conducted in terms of GSD and multi-site development. This also discusses challenges and opportunities in the DSD domain. The second area is RE, which plays a vital role in the software development process and the challenges of the DSD domain. In addition, this section

examines the customisation requirements of PS products, as presented in section 2.3, while section 2.4 presents a review of the adoption of Agile software for a DSD project, including the enhancement of locality and close contact to mitigate the challenges of DSD as well as RE issues. The third section reviews organisational boundaries and multi-site development. Finally, Section 2.5 discusses the overlap of the three areas in Figure 2.1 the DSD, RE and Agile areas to discuss local negotiation and decision making in the customisation requirements of distributed development projects. This section describes the importance of this scope and the recommendations made by different researchers to conduct more empirical research to discuss the challenges of customising PS products in the distributed domain and to focus on its requirements, as most literature research deals with this issue from the perspective of bespoke products, although there is a difference between the two, as seen in section 2.5. In conclusion, a summary section presents the main challenges discussed and the outputs are shown in Table 2.2.

2.2 Distributed Software Development Review

Until the 1980s, when the DSD concept began (Sengupta, Chandra & Sinha, 2006), the industry relied on the traditional method of in-house software development (Colomo-Palacios et al., 2014). There has been significant attention paid to DSD over the two decades since the 1990s (Kiani, Šmite & Riaz, 2013). Software development may be defined as the activities, methods, practices and technologies used by companies and individuals (Pressman, 2010). These are performed in the context of projects through work organisation teams, and projects may take different forms. They have a collocated form when the development team allocates a single site (Damian & Moitra, 2006) or they may be geographically distributed, where the actors in the software development process (e.g. project team, customer or user) are physically distant from each other (Prikladnicki, Audy & Evaristo, 2003).

The software industry now tends to decentralise and distribute both development and maintenance around the world in a phenomenon called GSD (Sengupta et al., 2006). DSD, including GSD, has received much attention owing to the importance of increasing distributed development and outsourcing projects (Schneider et al., 2012; Jiménez et al., 2009). The reason behind the tendency is the urge to allocate development teams to different distributed locations to obtain higher quality products and services at a lower cost, looking for skilled human resources and an appropriate infrastructure (Motira & Herbsleb, 2001; Khan, Niazi & Ahmad, 2009). According to Sengupta et al. (2006), the demand for global development arose in the late 1980s with copious research into the optimum practices for each type of software development. However, shifting from collocated to distributed projects has involved challenges as well as benefits, and these are of significant interest to more recent research (Sengupta et al., 2006).

DSD projects in the software industry are of different types, depending on variables such as the location and relationship of the various actors in the software development process (Robinson & Kalakota, 2004). For example, some projects involve many teams in a single city, whereas others have teams spread across different countries (Prikladnicki et al., 2007). The following section presents different forms (models) of software development practices in DSD projects and their challenges, with a glance at the research conducted into each.

2.2.1 Business models of distributed software development projects

Software development projects involve both the contract and the relationship between software vendors and client (Colomo-Palacios et al., 2014). These factors are of significance to DSD, shaping the interchange (Pressman, 2010), for example their geographical location, the distance between the distributed sites, the organisational structure and also technical aspects such as the development process, project structure and architectural strategy of each project (Pressman, 2010). Thus, business models of software development, particularly DSD projects, are defined by their relationship or location, or both. According to Robinson and Kalakota (2004), the relationship between the software vendor and client in software development projects may be of two types, depending on contract type and geographical distance, as shown in Figure 2.2. However, any project must involve both contracting and geographical distance models, as shown below and explained in the following sections.

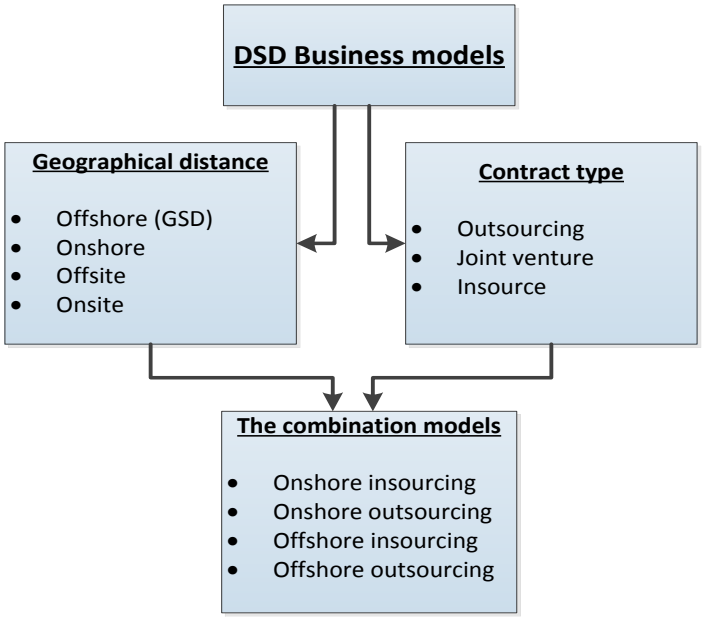


Figure 2.2: Business models in the DSD domain

2.2.1.1 Business models based on contract type

Developing software products for an organisation may be undertaken by the software development team in the organisation itself or by an external venture (Prikladnicki et al., 2007). In each case the relationship between client and development team depends on the type of contract involved: outsourcing, insourcing or joint venture (Prikladnicki et al., 2007).

I. **Outsource:** Outsourcing in some cases is known as external vendor or external buy. It involves project contracting and implies a faster process, often undertaken by companies that share offshore secondary functions (Prikladnicki et al., 2007). The concept started in the 1980s when IT firms tended to delegate development activities to specialist organisations with a large pool of skilled labour (Sengupta et al., 2006). There are many reasons for outsourcing software development such as low cost (Khan, Niazi & Ahmad, 2009) and the enhanced quality of both product and service achieved by using highly qualified practitioners (Shao & David, 2007). This kind of contracting in the software development industry has made a significant shift in recent decades and much research has been conducted to discuss its different issues (Oza et al., 2006; Tiwana & Keil, 2004; Damian & Moitra, 2006) and the challenges (Jiménez et al., 2009). Oza (2006) conducted intensive research into the relationship between software vendors and clients in Indian companies. This discussed various aspects of outsourcing such as the types, difficulties, benefits and success factors. Indeed, this kind of development model is the most challenging in DSD as it involves dealings with different organisations, so many issues arise in terms of organisational and cultural boundaries, and collaboration (Schneider et al., 2012).

II. **Insource**: Insource is another relationship between software provider and client, defined as business activities undertaken within the same organisation, even if these take place at different geographical locations (Schniederjans, Schniederjans & Schniederjans, 2005). It is also known as a subsidiary relationship in which development processes are executed, or a third party is brought in to work, within the client's organisation to build a software product and provide services (Prikladnicki & Audy, 2010). Although this kind of development models has fewer challenges than outsourcing it still confronts problems, especially when development team members are spread across different countries. In this case there are different time zones, communication and collaboration challenges (Höfner & Mani, 2007). Šmite et al. (2013) provide an example of research into insourcing development. In their study they explore offshore insourcing decisions through an empirical study and propose a decision-making process to help managers during software development.

III. **Joint venture:** This type of software development model refers to two or more software companies joining as a single entity to provide a software product. Sometimes foreign companies use a local company as a partner (Krishna, Sahay & Walsham, 2006). Tidd and Izumimoto (2002) conducted an investigation to examine 12 manufacturing joint ventures in the UK. They refer to this type of contracting as collaboration to exchange knowledge and experience in order to produce products, and this applies to the software production line.

2.2.1.2 Business models based the geographical location

Distributed software projects may be based on the geographical location of their actors (Robinson & Kalakota, 2004). DSD projects allocate team members to different locations during the development life cycle, and the distance between them may range from a few metres in the same building to across borders (Jiménez et al., 2009), when it is known as GSD. There are two models of software development process in this context, offshore and onshore (Prikladnicki et al., 2007), which take different forms depending on the type of contract and relationship with the software vendor. These are offshore outsourcing, offshore insourcing, onshore outsourcing and onshore insourcing, as shown in Figure 2.3 and explained below.

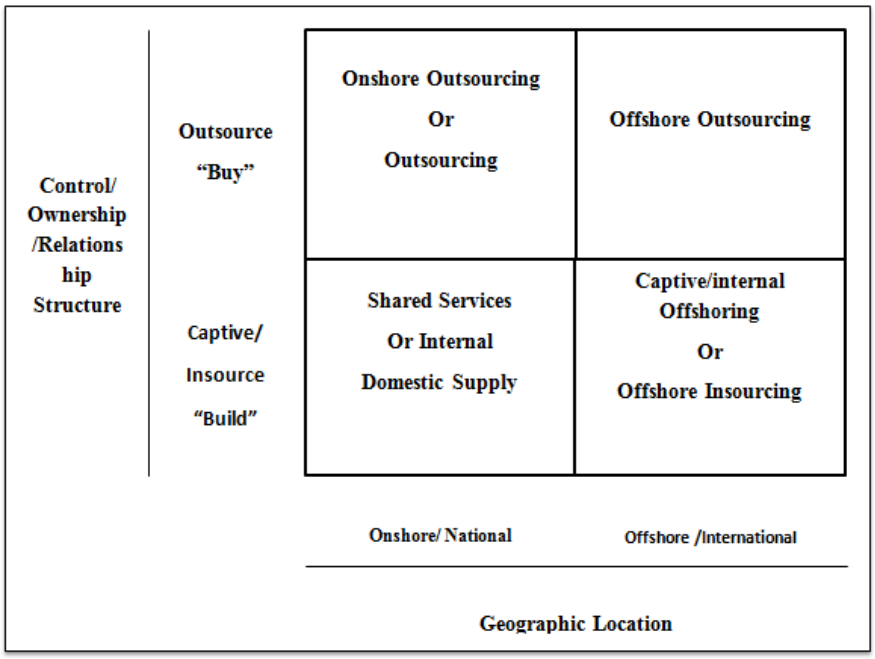


Figure 2.3: Matrix of DSD business models (Prikladnicki et al., 2007)

I. **Offshore:** Offshoring is a software development model where the client's and vendor's companies are located in different countries (Šmite et al., 2013). The development may be implemented by the contracting company or another vendor. This type is sometimes known as GSD (Noll, Beecham & Richardson, 2010) and is popular in the industry as it aims to manage the interaction between people, organisation and technologies comprising

development projects across international boundaries (Ebert, 2012; Herbsleb & Mockus, 2003). Much research has been undertaken on GSD, such as by Carmel and Tjia (2005), Khan, Niazi and Ahmad (2011), and Robinson and Kalakota (2004). The reason for this research is the significant demand for offshoring in the software industry (Hernández-López et al., 2010). Also, GSD has many challenges that demand investigation such as the time zone problem (Motira & Herbsleb, 2001), different cultures (Damian & Zowghi, 2003) and issues of development across organisational and border boundaries (Damian, 2006). There are many issues that need investigation such as the impact of GSD and offshoring development on team members (Colomo-Palacios et al., 2014; Bosch & Bosch-Sijtsema, 2010), indicating three important trends of GSD research: the widespread adoption of software product lines; the broad globalisation of software development across many organisations; and the importance of building a software ecosystem.

Offshore projects may be run as a form of outsourcing where the software product is provided by a subcontractor located in a different country to the client. Offshore outsourcing development is referred to as a modern business strategy for producing high quality software at little expense, using third party teams in countries operating at low cost (Khan, Niazi & Ahmad, 2011). This type of development integrates the contract-based relationship model of development (outsourcing) with a geographical development model (offshore). Since the 1990s there has been a significant revolution in the adoption of outsourcing software to offshore vendors (Khan, Niazi & Ahmad, 2011). There are many reasons, such as clients looking for cheap yet high quality software products (Bush, Tiwana & Tsuji, 2008). Outsourcing contracts to an offshore development team can satisfy this requirement, especially in countries such as India, Ireland and China (Khan, Niazi & Ahmad, 2011). According to McLaughlin (2003), developing software with offshore vendors in such countries usually costs only a third of that with onshore vendors and a half as much as in-house development, indicating the benefits. However, development projects using offshore outsource models face many challenges, especially in managing the coordination, communications, time zones and differences in organisations and cultures (Damian et al., 2007; Holmström et al., 2006). Therefore, balancing the benefits and risks in this type of research is vital to outsourcing and offshoring decisions (Holmström et al., 2006). The other form of offshoring is conducting projects by insourcing, when the company creates its own software through branches located in different countries. This type of development has come about as a result of globalisation, leading software organisations to undertake a development project across their sites. Most projects are run by multinational organisations that distribute their software development processes worldwide, seeking low-

cost, high quality labour to increase their customer base in different markets (Prikladnicki, Audy & Evaristo, 2004).

II. **Onshore:** Onshore is the form of distributed development project when both client and vendor are located in the same country (Prikladnicki et al., 2007). This kind of development model has fewer challenges than the offshore model, since factors such as culture, language, and time zones are similar within onshore development teams. However, projects still face difficulties in communication, collaboration and management, especially for multiple teams in different cities and in different organisations (Sengupta et al., 2006). Moreover, since most research has been conducted on offshore development (Khan, Niazi & Ahmad, 2011), there is a lack of studies focusing on onshore development projects. The onshore development model has two scenarios based on the location of the team and the site. In the first, the development process takes place in the same country but in a different city or venue (Sengupta et al., 2006); in the second, it is implemented at a single location: onsite at the client's premises (Robinson & Kalakota, 2004).

Onshore development projects are known as outsourced when an external company (subcontractor) located in the same country provides software products or services to a client's organisation (Faiz et al., 2007). Outsourcing the process usually indicates savings in the cost of qualified labour, but there is little justification for organisations to outsource to local software development firms (Laplante et al., 2005). Some researchers attribute the reason to avoidance of the challenges and difficulties of global development across borders, such as different languages, time zones and cultures (Carmel & Tjia, 2005).

Another onshore development model is the insourcing of software development, when the software is developed through internal projects (in the client's organisation) in the same country. In this model the company takes the development role and produces software for the organisation itself, using its resources by delegating the responsibility for development to one of the firm's departments (Iqbal, Ahmed & Marczak, 2014). This type of development is referred to as a simple development model since the challenges of offshoring and outsourcing projects are absent, yet there are other issues such as a lack of qualified labour if the development is undertaken within the same organisation (Prikladnicki et al., 2007).

2.2.2 Shifting from collocated to DSD and trend of distributed research

The software industry has seen a significant leap forward in various areas in recent decades, and one of these areas is DSD, a growing field in software engineering (Prikladnicki & Audy, 2010).

Many companies have distributed their software processes and tend to decentralise and distribute development and maintenance around the world, looking for competitive advantages in terms of quality, cost and human skills resources (Ranganathan & Balaji, 2007). The DSD phenomenon started in the late 1980s (Sengupta et al., 2006), and some authorities would defer that to the early 1990s (Carmel & Tjia, 2005). However, the most significant interest and intense competitive research has taken place since the 1990s in an increasing number of DSD projects, whether local, global, within same company (insourcing) or with a third party (outsourcing) (Schneider et al., 2012; Jiménez et al., 2009). Furthermore, in 2001 the Agile software development approach began to be adopted by the software development field after the announcement of the Agile Manifesto (Fowler & Highsmith, 2001; Kent Beck et al., 2001), changing the software development process from traditional to Agile. This played a vital role in the increase in demand for DSD and distributed Agile development (DAD) research (Sureshchandra & Shrinivasavadhani, 2008; Jalali & Wohlin, 2010).

The transition from co-located to distributed projects has been accompanied by significant changes to aspects of the software development field (da Silva, França & Prikladnicki, 2010). These include features and challenges integral to software development project practice (Jiménez et al., 2009). Much research has discussed the DSD phenomenon from different angles such as software development environments, including development centres (Carmel & Agarwal, 2006; Höfner & Mani, 2007), project management (Ribeiro, Czekster & Webber, 2006) and project allocation decisions (Ebert, 2007). People involved in DSD projects have taken a unique interest in research into the relationship between vendor and client, the impact of DSD on distributed team members and cultural differences, including language (Colomo-Palacios et al., 2014). Tools and technologies have also been investigated to provide alternatives to close contact or to overcome the challenges of DSD projects (Rodríguez, Ebert & Vizcaino, 2010). Although these aspects have been covered in the DSD research literature, a significant number of studies incorporate GSD issues such as communication, collaboration, time zones in global development, cultural and organisational boundaries in outsourcing projects and managing requirements over distributed domains. The following sections in this chapter present the challenges of DSD and propose solutions in terms of models and frameworks in DSD including GSD, and Chapter 3 presents a framework for the challenges of DSD customisation with details of each challenge. The issues identified in that chapter have been verified by various expert practitioners through a questionnaire.

2.2.3 Distributed software development challenges

DSD, including GSD, has brought significant benefits to the process of development and accelerated software production. However, many challenges have arisen in the wake of global

development, such as inadequate communication between teams and problems with time zone differences (Sengupta et al., 2006). In recent years, a number of studies including systematic reviews have been carried out to discuss DSD challenges, issues and difficulties.

For instance, Jiménez et al. (2009) conducted a systematic review of the literature relating to the challenges and issues of DSD. Their study proposed solutions and ways of meeting these challenges. They addressed the challenges of DSD projects such as communication, group awareness (relationships between people in the project), configuration management, knowledge management, coordination, collaboration, project and process management, process support, risk management, cultural differences and quality measurement, proposing how to meet each challenge.

Another study was initiated by Sengupta et al. (2006) at the IBM Research Centre to investigate the challenges of DSD. This identified four issues: collaborative software tools, knowledge acquisition and management, testing in a distributed set-up, and process and metrics. It addressed the difficulties in each area and the research gaps such as inadequate communication, trust, system integration and knowledge management. The authors investigated RE issues in DSD, especially across cultural boundaries and in stockholder organisations, and were able to construct a model of the requirement-gathering process, including negotiation and specification, showing the difficulties faced by software development in DSD projects in terms of RE. Rodríguez, Ebert and Vizcaino (2010) conducted a study to investigate the tools and technologies used by distributed teams. They discussed the collaboration and integration of these technologies and the tools involved in software processes, such as IBM Jazz, Microsoft SharePoint and Google Apps. Their study included a comparison of these technologies with the software development process, such as tracking systems and management features, plus calendar management.

Another DSD aspect investigated in the literature is multi-site development and organisational boundaries. Software development processes such as RE and project management are difficult enough in a local development setting but become even more so when multi-site projects are involved. This problem increases over organisational and cultural boundaries. A great deal of DSD research has discussed the impact of multi-site projects on the development process itself, as well as organisational boundaries, for instance the study by Damian and Zowghi (2003). This identified generic challenges such as inadequate communication, knowledge management, cultural diversity and time differences. In the same context, Damian (2007) considered the challenges and difficulties of global software engineering (GSE). The study states that managing the requirements of distributed stockholders in software development in the DSD domain across cultural, time zone and organisational boundaries faces significant issues. In

addition, it shows how DSD challenges such as communication and coordination become more difficult in multi-site projects spanning organisational and cultural boundaries.

2.2.4 Proposed solutions to overcome DSD challenges

DSD challenges have been the focus for researchers and practitioners (da Silva, França & Prikladnicki, 2010) and many studies have attempted solutions (Portillo-Rodríguez et al., 2012). Studies adopt one of two approaches. Some focus on technical problems and propose technical remedies for issues such as inadequate communication, poor bandwidth and connectivity problems (Portillo-Rodríguez et al., 2012). Many of these discuss collaborative software tools to reduce the challenges involved in a DSD project. Rodríguez, Ebert & Vizcaino (2010) carried out a study focusing on tools and technologies designed for distributed teams, discussing a number of tools in the software market, as summarised in Table 2.1. In their study they indicate the power of the support provided by these tools to communication, coordination and control. However, these technologies do not cover all phases of the software development life cycle. Another study discusses tools and technologies for different DSD aspects such as RE (Carrillo de Gea et al., 2012; Teruel et al., 2014), software testing in the DSD domain (Andrea, 2007) and quality (Lanubile et al., 2003; Martignoni, 2009). However, the proposed tools and technologies focus on communication and management; few studies look into tools that support coding over distributed boundaries, representing a gap in the research that needs more investigation (Sengupta et al., 2006).

The second approach, taken by many studies, is to discuss and propose solutions to DSD challenges by focusing on business aspects such as models and frameworks to deal with particular difficulties. Schneider et al. (2012) conducted a systematic literature review to collect and analyse 127 solutions proposed for four aspects of DSD and GSD: process management; project management; engineering; and support. Project management has received most attention and the other areas such as product integration very little, and the study concludes by identifying those that need more focus: project monitoring; product control; and integration. These recommended areas could be taken as a pointer to future research in the DSD domain. Šmite and Borzovs (2006) investigated the impact of risk management on GSD, termed global risk, designing a framework to address this across organisational and cultural boundaries. Akbar et al. (2008) proposed a model for those software companies developing web applications for distributed client locations that emphasises the support needed for communication between developers and offshore clients to complete their projects with the minimum documentation (Figure 2.4).

Table 2.1: Collaborative technologies for distributed teams (Rodríguez, Ebert & Vizcaino, 2010)

Collaborative technology	Technology tools	Data backbone	Compatible tools	Web 2.0 apps	Security
IBM Jazz (www.01.ibm.com/software/rational/jazz)	Rational Team Concert, Rational Quality Manager, Rational Requirements Composer, Rational Asset Manager, Rational Insight	Integration based on Open Services for Lifecycle Collaboration using Jazz Foundation Services across independent databases that use Internet protocols	Rational ClearQuest, Rational ClearCase and Subversion, Lotus Connections, Microsoft SharePoint	blog	Lightweight Directory Access Protocol (LDAP)
Microsoft SharePoint (http://SharePoint.microsoft.com)	SharePoint Sites, SharePoint Communities, SharePoint Content, SharePoint Search, SharePoint Insights, SharePoint Composites, SharePoint Designer, SharePoint Foundation, SharePoint Workspace	Windows Share-point Services Data Integration through platforms such as Mamboo MashPoint or Mainsoft Share-Point Integrator	Microsoft Office, Lotus Notes, Domino	wiki, blog	Secure Sockets Layer (SSL), IP Security, Windows Security Account
Google Apps (www.google.com/apps)	Google Calendar, Google Docs, Google Groups, Gmail, Google Sites, Google Videos	Data Integration through Google Secure Data Con-nector	Microsoft Outlook, Lotus Notes	wiki, blog, video	HTTPS, SSL, Message Security through Postini technology
IBM Lotus (www-01.ibm.com/software/lotus)	Lotus Notes, Lotus Domino, Lotus SameTime, Lotus Live, Lotus Quickr, Lotus Connections	Data Integration through Lotus Enterprise Integrator	Rational Team Concert	blog, wiki, microblog	SSL, Access Control List (ACL)

Mudumba and Lee (2010) found that there was a lack of research into DSD risk management. As a result, they proposed an Agile risk management framework that supports identification of dynamic risk management. Interestingly, they discussed multi-organisations, multi-teams and other multiplicities in DSD, and reported that several researchers recommend this Agile method in project management to mitigate dynamic risk in software development projects.

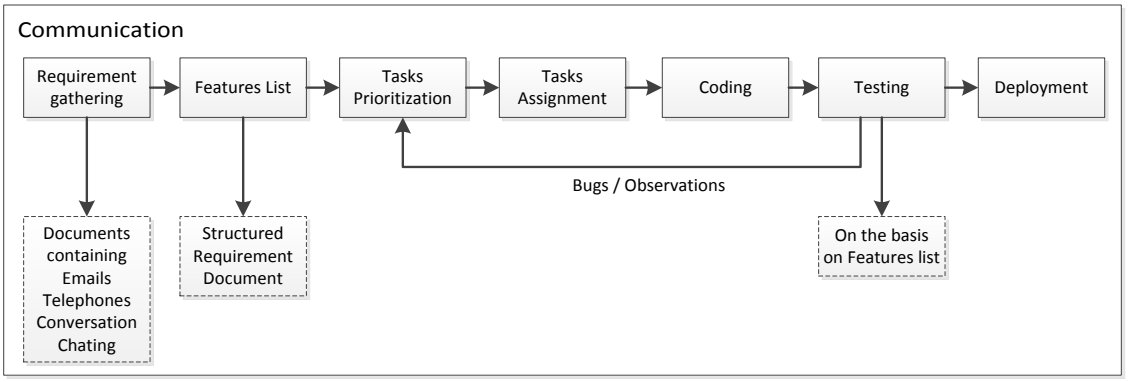


Figure 2.4: The DAD model (Akbar et al., 2008)

In short, the previous section shows the importance of DSD and presents different models of DSD projects in both the global and the local domains, discussing research trends and proposing solutions to some challenges in the literature. The following points represent the outputs of this section:

The software development industry, both in practice and research, focuses on dealing with the challenges and opportunities of the distributed domain. Thus, researchers are prompted to conduct studies in this area to fill the gaps.

Communication among the distributed teams is one of the main challenges facing any DSD project in all the models. However, the level of challenge depends on the degree of distribution; for example, onshore insourcing projects face fewer challenges than offshore outsourcing projects. Therefore, the literature encourages solutions comprising a combination of different models.

Much research emphasis is on the importance of research conducted in the real world, using empirical data to deal with issues and problems with realism, proposing solutions that meet both the software industry's and practitioners' needs.

2.3 Requirements Engineering in Distributed Software Development Review

Requirements are defined as a specification of what should be developed and implemented in order to produce a software product (Sommerville & Sawyer, 1997) and RE refers to the process of dealing with software requirements, including their formulation, documentation and management (Kotonya & Sommerville, 1996). It is a critical phase in software engineering (Damian, 2002; Jiao & Chen, 2006), with a vital role in managing customer requirements, a principal factor in production in the marketplace (Niazi et al., 2012). Since the 1990s the software industry has shifted its attention to GSD, as explained earlier in this chapter. However, to meet the challenges, changes in software engineering practices are needed (Damian et al., 2004), and specifying requirements is one of those practices affected. While it is difficult to manage these requirements locally, it is even harder to communicate them over organisational boundaries and to distribute them to multiple customers (Damian, 2002). This difficulty increases in DSD projects as well as those with multiple stakeholders across organisational and cultural boundaries. In recent years, a significant transition from co-located forms of development to GSD has taken place that demands collaboration (Damian and Zowghi, 2003; Damian, 2007), and in a distributed setting RE is a complex intersection phenomenon that involves technical, social and organisational aspects (Damian, 2007).

2.3.1 Customisation requirements for packaged software products

The software industry provides different types of products and processes. Some products are built from scratch, based on clients' requirements, whereas others are sold 'off-the-shelf' and customised. Therefore, the RE process is differentiated by software type and the extent of changes to the product agreed between vendor and client. Most research that discusses the development process, practices and management of development team roles and responsibility, including requirement management during the software development life cycle, focuses on the development teams responsible for dealing with client requirements, from selection to delivery of the solution through a process of decision making and development.

Xu and Brinkkemper (2007) indicate differences between the following software products, as shown in Figure 2.5.

- **Shrink-wrapped software**: Software that is boxed, shrink-wrapped then sold in stores to run on different platforms. In this type, the user has no option to change the function.

- **Commercial off-the-shelf software (COTS)**: COTS software built for the general public in the software market. It can be used as it is, or have some features personalised (for example, the application's appearance) without changing the main functions. This type of software is sold without source code modification (Brownsword, Oberndorf & Sledge, 2000).

- **Commercial software:** Products purchased in the retail market with a strict licence for use, without permission to change or copy.

- **Packaged software:** PS is defined by Xu and Brinkkemper (2005) as ready-made software products obtainable from software vendors that are built to general requirements then sold to different clients, and that may be ordered with the source code then customised to the client's requirements. Examples of this type of software product are Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems. Over the past decade a number of organisations have decided to implement PS, so it is one of the fastest-growing software industry markets (Jebreen & Wellington, 2013). In addition, Karlsson et al. (2007) mention a study conducted by AMR in 2007 that refers to the importance of the PS market and how it was expected to undergo significant growth between 2007 and 2009 to $64.88 billion. However, the PS domain and market need more research to identify gaps in our knowledge and other challenges. For instance, the PS market needs to identify the difference between the development of packaged software and that of bespoke software (software built from scratch according to a client's specification) (Sawyer, 2000).

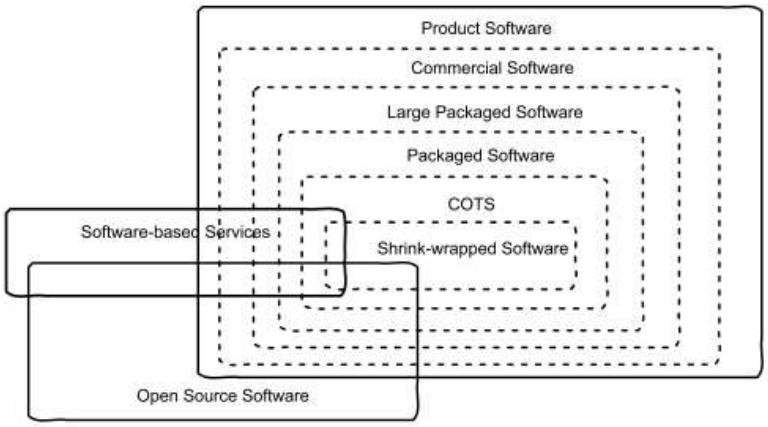


Figure 2.5: Categories of software products (Xu & Brinkkemper, 2005)

The trend during the previous decade has been for the software development market to pay significant attention to the specialisation of software products in terms of custom-ready software, rather than building software products from scratch, in order to save time and to deliver a high quality and trusted product (Sawyer, 2001). In addition, custom-ready software such as software packages has attracted interest as it has wide-ranging effects on the

configuration of the organisation, so software vendors have focused on how to build this type of software that is packaged and customised to meet clients' requirements, while client organisations have focused on how to manage the environment for that software, including the people and hardware to help to make the product work by defining accurately the requirement (Jebreen, 2014). Sawyer (2001) also considers the change in the RE processes dealing with the customisation requirements of different clients. The following sections discuss the challenges of RE, including customisation requirements and opportunities in DSD projects.

The fast growth and great significance of PS, and the seeming predictability of its developing process, make it necessary to understand RE practices in packaged software implementation and the development process (Light, 2005). However, Jebreen and Wellington (2013) emphasise the absence of studies discussing RE practices for PS products. The trend of future research is toward the different aspects of RE, given the lack of RE tools to support the analysis and management of, and collaboration with, clients' customisation requirements for PS products in the distributed domain (Jebreen & Wellington, 2013).

2.3.2 Challenges of RE in the distributed domain

During recent decades the amount of research and the number of publications in the fields of RE and requirements management have increased (Jiao & Chen, 2006). Nowadays, software is becoming complex and has a number of requirements to meet clients' goals, and the software development process has paid attention to RE and RE research (Pitangueira, Maciel & Barros, 2014). In RE, the main goal is to identify and meet the client's requirements to satisfy the demands of stakeholders (Zhang & Harman, 2010), so the human factor has an important role involving challenges such as decision making, prioritising clients' requirements, estimating the project budget and duration, and dealing with different kinds of customers (Pitangueira et al., 2014), as its success lies in understanding and meeting the needs of users, customers and stakeholders (Cheng & Atlee, 2009). In addition, other aspects of RE in the distributed and global development environment have been discussed in the literature in order to identify challenges and propose solutions (Schmid, 2014): different aspects of RE such as requirements elicitation (Sharp, Finkelstein & Galal, 1999; Cohene & Easterbrook, 2005); requirements modelling (Alfonso, Braberman & Kicillof, 2004; Alur, Etessami & Yannakakis, 2003; Damas, Lambeau & van Lamsweerde, 2006); requirements analysis (Campbell et al., 2002); and requirements management (Jiao & Chen, 2006; Khan, Basri & Dominic, 2012).

In the context of the DSD domain, Damian and Zowghi (2003) conducted research focused on the challenges facing RE in the distributed domain. They proposed a model, shown in Figure 2.6, presenting the major problems behind the geographical distribution of stakeholders:

cultural diversity; inadequate communication; knowledge management; and, in the case of GSD, time differences. Moreover, they indicated corresponding challenges such as a lack of informal communication of the local context, difficulty in understanding requirements, ineffective decision-making meeting and delays in communicating requirements across distributed boundaries.

Other studies have discussed the challenges of communicating requirements of different stakeholders in the GSD domain, such as problems in knowledge acquisition and knowledge sharing, effective communication and coordination between teams (Damian, 2007). These challenges are confirmed by studies such as that by Gopal et al. (2011) as major challenges still facing RE in DSD projects. This study discussed communication and coordination issues in the distributed domain, and the impact of these challenges on the speed and quality of the software service for multiple distributed clients. It examined internal and external teams in distributed projects and found that close contact between clients and development team reduces the impact of the challenges to communication and coordination in DSD projects. Schmid (2014) discussed challenges of RE in the GSD domain such as organisational and communication issues in customer requirement selection, cultural diversity and requirements negotiation and prioritisation. The study concluded by emphasising the role of RE in DSD issues, also the importance and need for research in various aspects such as communication, negotiation and the prioritisation of clients' requirements across distributed boundaries (Schmid, 2014).

In terms of the RE challenges for PS products, Dahlstedt et al. (2003) indicated a difference between PS and bespoke software projects. Subsequently, Xu and Brinkkemper (2007) identified issues faced by PS development processes such as requirements, implementation and delivery. They pointed to the pressure faced by PS vendors relating to 'time to market'. PS frequently deals with clients' requirements through cumulative releases to adjust software functions and this process requires careful planning, prioritisation, testing and management of control, particularly when dealing with different customers and in receipt of multiple customisation tasks (Xu & Brinkkemper, 2007; Karlsson et al., 2007). Other studies in the literature have discussed the requirements challenges of PS projects. For example, Mishra and Mishra (2009) conducted research into the impact of Agility on RE and suggested that Agile methods should be used to deal with PS to deliver customisation requirements incrementally, as the development approach handles 'time to market' challenges by delivering incremental versions according to priority.

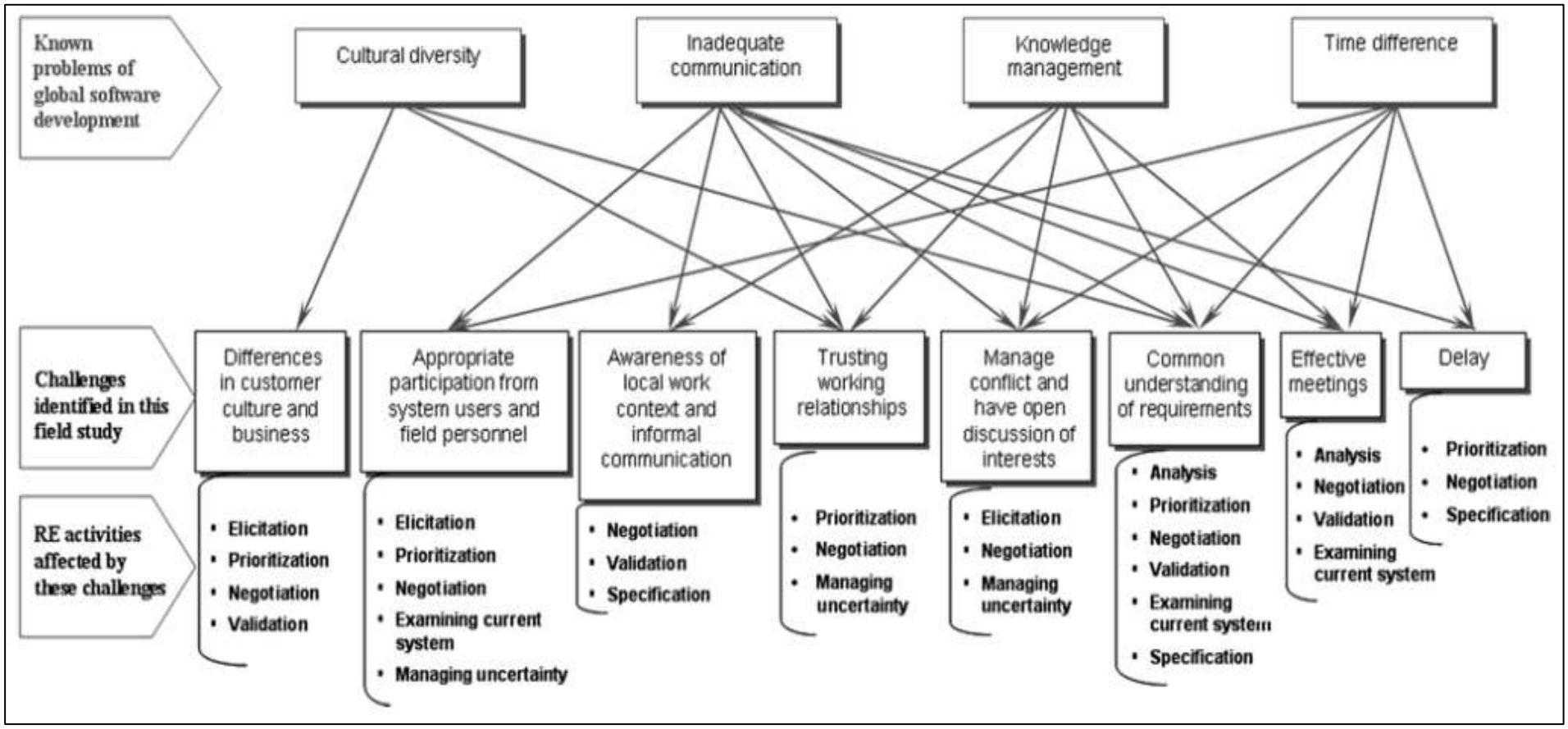


Figure 2.6: A model of challenges aspects of RE in DSD domain (Damian & Zowghi, 2003)

2.3.2.1 Proposed solution for RE challenges in DSD projects

Distributed stakeholders are the main challenge for RE in both the development and customisation of software. Damian (2007) presents different lessons that have been learned from practice, with a focus on global development projects across organisational and cultural boundaries. Moreover, this study discusses how much knowledge acquisition and sharing between developers and stakeholders would help RE in the GSD domain. In the same context of RE for GSD, Hayat et al. (2010) proposed a model to manage requirements changes during the development process. Next, Khan, Basri and Dominic (2012) extended the model by proposing a framework (RCM) for managing requirements changes in the GSD domain.

The proposed framework also emphasises knowledge sharing by adding a new component to the framework, a 'central repository' that makes any change to one customer's site visible to other customers. The authors are to evaluate this framework by conducting case studies in different small- and medium-sized GSD organisations (Figure 2.7). Another study has been conducted by Damian and Zowghi (2003) on the impact of distributed stakeholder on requirements management. They proposed a model to address the impact of factors such as remote communication and knowledge management, cultural diversity and time differences on requirements management in distributed development projects. They concluded their study with recommendations for practice in DSD environments to address the main challenges. To manage and prioritise requirements in a distributed development environment, Ahmad et al. (2011) conducted two experiments to discuss the challenges and attributes of prioritisation requirements in the Agile development process in the GSD domain. They concluded by emphasising the importance of prioritising requirements, for developers as well as customers.

To sum up, RE plays a vital role in the software development process and software engineering in general. During recent decades the research has gone through different stages and has interesting RE aspects, including managing customers' requirements at collocated projects for developing bespoke software products, RE for Agile software development project, clients' requirements in customisation projects of PS products, and RE in distributed development projects including multi-site and GSD projects. Much research has been conducted into those aspects. However, recent research such as by Mishra and Mishra (2009), Jebreen and Wellington (2013), Gregory and Baskerville (2014), and Jebreen (2014) emphasises the lack of research, as research has discussed RE in PS projects in collocated and distributed projects. Also, these have identified challenges in RE in PS projects, with emphasis on the need for more investigation, real world studies and developing tools to deal with PS requirements, especially in the DSD domain, as the existing tools and models focus on bespoke products, and these are unlike PS products.

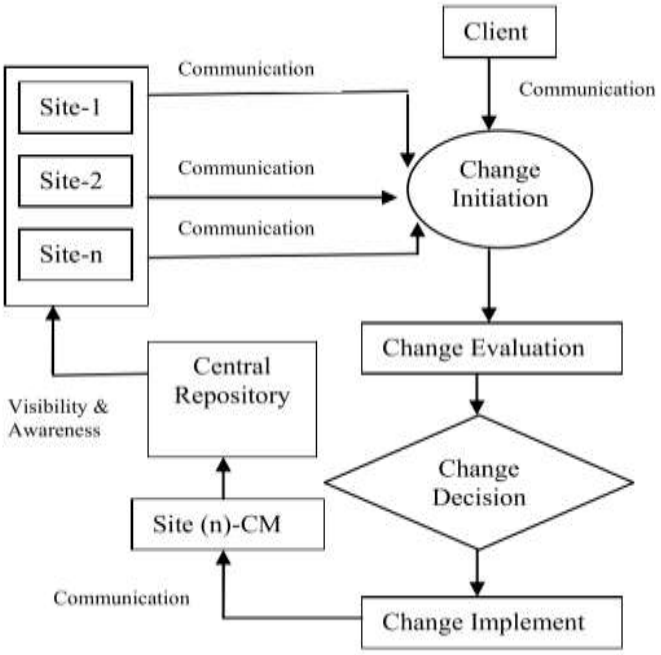


Figure 2.7: RCM framework for GSD domain (Khan, Basri & Dominic, 2012)

2.4 Agile Methods and Locality in Software Development

Agile software development is 'a phenomenon' (Dingsøyr, Dybå & Abrahamsson, 2008), not merely a development approach or methodology. It is a philosophy of software development and a new way of thinking about development processes and project management (Shore & Warden, 2007; Fowler, 2001). The business community (Abrahamsson et al., 2002) demands a development method that is lighter and faster than the traditional plan-based approach. It has arisen in reaction to traditional models, such as the waterfall model, to reduce development time and costs as well as to accommodate changes in requirements at any time with no detriment to the duration of the whole development. As a result, the Agile method is a phenomenon in the software development process and community (Cohen et al., 2004; Dybå & Dingsøyr, 2009). According to Shore and Warden (2007), 'Agile methods are processes that support the Agile philosophy'. Schwaber and Beedle (2002) describe the Agile method as an approach to manage the system development process. Many Agile methods have been launched in industry, for example the Extreme Programming (XP) (Fowler, 2001), Scrum (Schwaber, 2004) and Crystal Clear methods (Abrahamsson et al., 2002).

Now DSD has become commonplace in modern software development, Agile software development is attracting industry attention, with a recent trend in distributed development

environments to adopt the Agile methodologies (Sureshchandra & Shrinivasavadhani, 2008). 'DAD', or distributed Agile development, refers to adopting Agile principles in DSD to secure the advantages of Agile software development and using distributed development projects. As Agile practices promote a development iteration process through Agile methodologies, this can help DSD to tackle its challenges such as differences in culture and communication (Phalnikar, Deshpande & Joshi, 2009).

However, there are many reports of organisations adopting Agile methods in distributed development environments in different forms (Lee & Yong, 2009). Jalali and Wohlin (2010) conducted a systematic review on using Agile methods in GSD and concluded, for instance, that the adoption of Agile methods in GSD is not well investigated. Although there has been a significant increase in publication in this area, mainly industrial reports and a few evaluation studies, there is a need to conduct in-depth studies discussing the challenges and benefits of applying Agile concepts in DSD projects (Jalali & Wohlin, 2010).

2.4.1 Locality in Agile software development

Agile software development methods rely on the concept of 'Agility' in development process (Morien & Wongthongtham, 2008). This stresses practices mentioned in the Agile Manifesto (Kent Beck et al., 2001), the keys factors of success in adopting the Agile approach for software development projects. These factors emphasise close communication with the development team, including the client (Abrahamsson et al., 2002), such as making decisions and negotiating in close contact, involving clients in the prioritisation process, sharing knowledge and collaborating with team members in close meetings such as the 'scrum meeting' in the Scrum method (Paetsch, Eberlein & Maurer, 2003; Moe & Aurum, 2008). Close communication in the Agile development approach provides the environment vital for negotiating software requirements with the client, developer and decision maker in order to make accurate decisions and prioritise the client's requirements (Cohen et al., 2004; Racheva et al., 2010). Doing this locally in close contact with the client enhances the software development process by reducing both the cost and duration of the development process (del Nuevo, Piattini & Pino, 2011), important features of a successful project (Jiang & Klein, 2000). Moreover, close negotiation of a client's requirements provides the development team with understandable and agreed requirements that help the productivity and quality of the software development process and result in software that achieves stakeholder satisfaction (Abrahamsson et al., 2002). Another factor with an important role in an Agile development project is eliciting and managing client's requirements using informal methods in an iterative process to produce incremental software products that meets clients' requirements and gains stakeholders' satisfaction (Cohen et al., 2004; Bjarnason, Wnuk & Regnell, 2011). In the Agile development approach the locality

provides a flexible environment for close discussion between clients, developers and decision makers, helping to reduce the challenges of formal communication besides enhancing knowledge sharing and collaboration among team members (Dybå & Dingsøyr, 2009). In addition, using the Agile approach in software development projects is popular in industry as it shows positive benefits in development teams at educational institutions, for example, in software engineering training courses (Joy, 2005).

In sum, all these factors of success in the adoption of the Agile approach in software development projects indicate the benefits of communicating closely with clients for both the quality and productivity of the software development.

2.4.2 Adopting Agile software development in DSD projects

Agile methods focus on factors of collaboration and establish the concept of close communication between the client and the development team (Erickson, Lyytinen & Siau, 2005). Furthermore, Agile methods rely on face-to-face and informal communication to deliver working software within time and budget constraints (Bowen & Maurer, 2002). Thus, adopting Agile methodology in DSD projects, where the parties are distant, faces many challenges (Jalali & Wohlin, 2010). Coram and Bohner (2005) examined the impact of Agile methods on software project management. Their study discussed the impact of applying Agile methods on the process of the project, as well as the people involved such as developers, testers, project leaders and customers. They also considered the management and development processes (e.g. planning and documentation). Some researchers have concentrated on a single aspect of project management, such as risk management, then examined the impact of distributed development or Agile development on that aspect. Fowler (2003) reported on his experience of adopting Agile principles in an offshore development project, discussing the importance of various factors in Agile development such as communication, cultural changes and documentation. He set out the challenges as well as the benefits of applying Agile methods in offshore projects, as well as the current and future trends, stating that ‘Offshore development is very fashionable’. In addition, most DSD challenges apply to DAD. Therrien (2008) presented the experience of First American Core Logic (FACL) in adopting Agile Scrum in DSD projects, describing challenges such as time zones, communication and cultural challenges, the trust and enablement factor and technical challenges such as the tools of communication. He concluded by recommending an improvement in the level of adoption of Agile methods in distributed projects.

On the other hand, Agile has held out many features for software development during recent decades, for both collocated and distributed development projects (Nisar & Hameed, 2004; Sulfaro, Marchesi & Pinna, 2007). Much research has discussed the benefits of adopting Agile

methodology in DSD including GSD, such as improving the productivity of development teams in DSD projects (Ribeiro et al., 2006; Hossain, Babar & Paik, 2009). Also, communication is referred to as a dilemma in DSD projects; Fowler (2003) stated that, 'Our experience is that, even if an Agile approach suffers from the communication difficulties of offshore, it's still better than a documentation-driven approach'. Another study confirmed that, like that of Kircher et al. (2001) and Layman et al. (2006), research conducted to investigate the use of Agile methods in DSD projects indicated it had benefits for communication, an informal communication environment and enhancing collaboration among teams in the distributed domain.

2.4.3 Requirements management in DSD projects under Agile development approach

Agile software development has become a significant movement in the history of software development (Abrahamsson et al., 2002). It has changed RE as well as development processes, making it more flexible and lighter than traditional approaches. In addition, Agile teams customise software products as well as implement changes with the benefits of closer interaction by producing integrated working software based on customers' requirements and priorities. A number of studies have discussed Agile development in terms of RE. Racheva et al. (2010) investigated requirements priorities from a user's perspective in the Agile development process. They proposed two conceptual models for prioritisation requirements in Agile software development (Figure 2.8 and Figure 2.9). To conclude, they raised questions about research gaps in terms of providing guidelines for researchers and practitioners in this domain, focusing on the techniques of Agile approaches to managing requirements in terms of prioritisation.

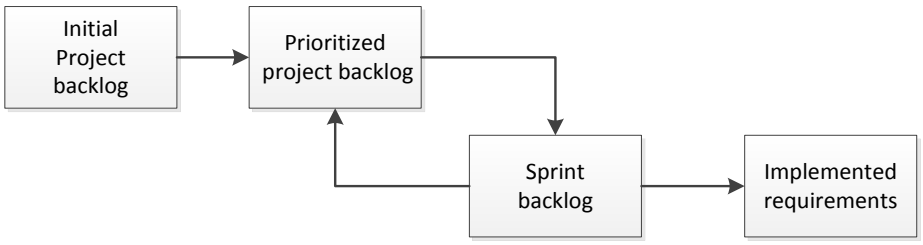


Figure 2.8: Prioritisation model for Agile requirements (Racheva et al., 2010)

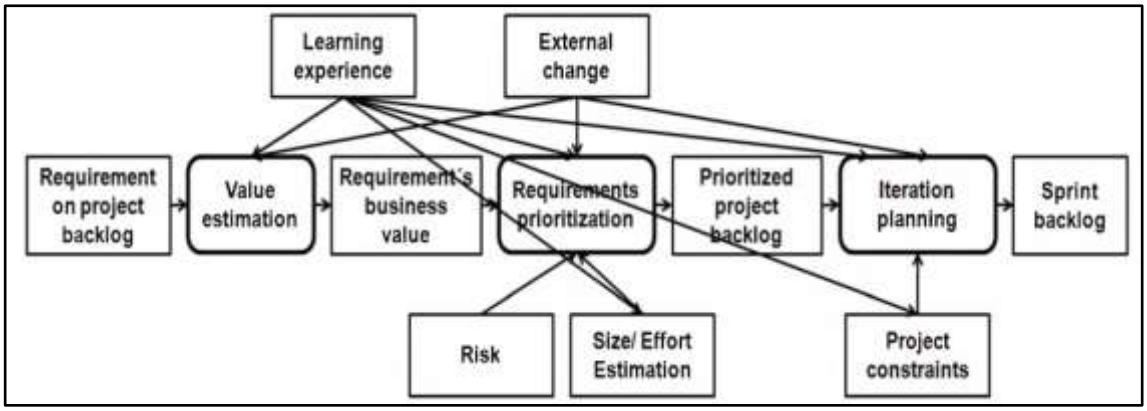


Figure 2.9: Conceptual model of managing requirements in Agile projects (Racheva et al., 2010)

Another related field is the use of an Agile method, such as Scrum, in distributed development projects at a customer's location, with benefits for requirements management and the development process in general. Vlaanderen et al. (2011) proposed an extension of the Scrum process, an 'Agile requirements refinery', to deal with major requirements in the development environment (Figure 2.10). They evaluated their work in a real-life case study. To conclude, they emphasised the lack of research in the area of Agile software management, including requirements management in the Agile development process.

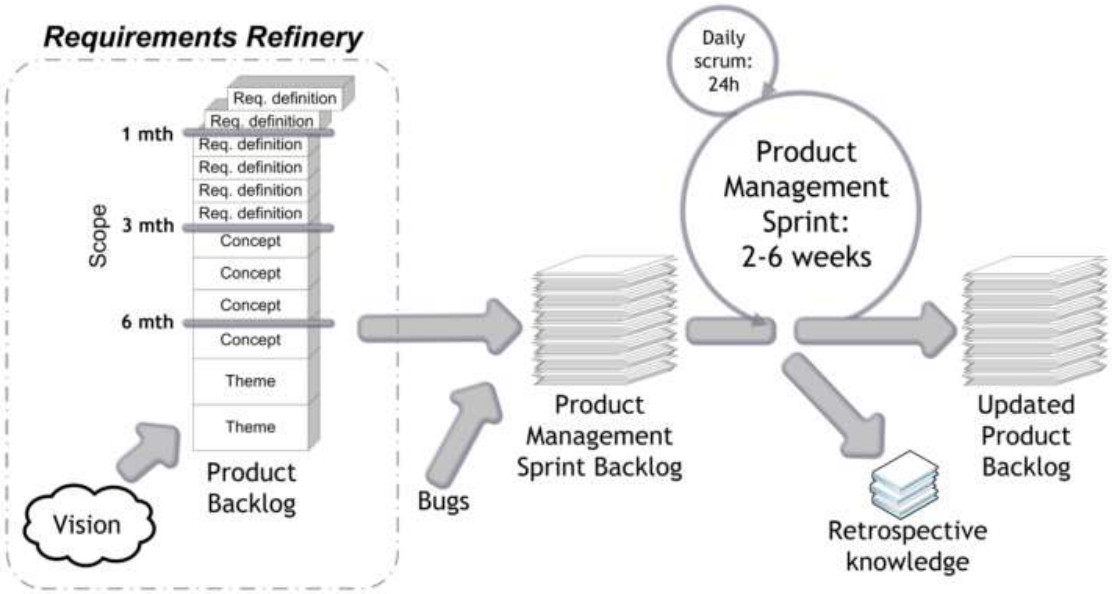


Figure 2.10: Agile software management model using the requirements refinery (Vlaanderen et al., 2011)

To summarise, the Agile software development approach has changed the software industry and enhanced new practices in software development, supporting by making lighter. This affects

both collocated and distributed development projects. The Science Agile approach relies on a less documented process with close communication among development team members in order to support requirement negotiation to make decisions and to prioritise with the involvement of clients. This style of Agile approach is applied to distributed projects to reduce the amount of formal communication between distributed teams. However, some challenges will continue to face projects as Agile methods are designed primarily for collocated teams, and extending their use to distributed teams will demand changes if they are to be useful.

2.5 Local Negotiation and Decision Making for Customisation Requirements of Distributed Development Projects

Since 2000 a number of phenomena and concepts have emerged in the software industry and market that have affected several aspects of software engineering, such as Agile software development methods (Erickson et al., 2005), an increase in DSD projects (Prikladnicki & Audy, 2010), and growth in using and producing PS products (Morabito, Pace & Previtali, 2005). Consequently, research has focused on the impact of these phenomena on different activities in the software development life cycle in order to identify and discuss the opportunities and challenges of this change. Figure 2.1 shows three different areas of software engineering relating to the development life cycle and projects, namely DSD, RE and locality, used in Agile methods.

DSD offers benefits to software development firms and the industry in general through many features, as discussed in section 2.2.2, such as using low-cost labour to produce high quality software products, saving time and money by delegating the development process on a multiple distributed ‘time to work’ parallel. Also, the software market has enlivened many developing countries in conjunction with software development firms. For example, software engineering emphasises building software in systematic way to provide high quality products with minimal cost (Chow & Joy, 2005). However, DSD projects involve challenges such as communication, collaboration, organisational and boundary issues (Jiménez et al., 2009). In contrast, Agile software development has seen the field change from a heavy, traditional process to lighter and incremental operation using practices such as close contact and local negotiation, decision making and prioritisation of project requirements with greater client and development team collaboration (Vlaanderen et al., 2011; Holmström et al., 2006).

RE plays a vital role in software development activities concerned with handling client requirements, a basic factor in the development production process in the software market (Niazi et al., 2012). RE has been referred to as an important potential influence on the success of

a development project, whether that project has a collocated or a distributed nature. Much research has discussed RE in the context of DSD and identified the key challenges of the overlap between RE and DSD areas (as discussed in section 2.3). In short, the main challenge of RE in the distributed domain is communicating clients' requirements across distributed boundaries, past different obstacles such as a contrasting culture, multiple organisations, difference and lack of collaboration. This problem exists in bespoke as well as PS products, as recent research identifies a gap in the research on customisation requirements in the distributed domain (Jebreen & Wellington, 2013). Jebreen & Wellington state that 'Future research could consider a problematic area within existing RE tools is that they do not support a distributed collaboratively collection and analysis of requirements'. Also, other research issues concerning customisation requirements have been referred to as an aspect of software development that requires more focus and discussion (Jebreen, 2014).

Research conducted in this area has discussed different aspects such as the prioritisation of requirements. Berander and Andrews (2005) considered different techniques of prioritisation, as it is vital to make decisions regarding cost and time by selecting from multiple alternatives. They indicated the importance of involving stakeholders in the prioritisation process, deciding which requirements are important and which need to be developed first. They concluded their work by emphasising the need for empirical research and real world cases on the prioritisation of requirements, using different prioritisation techniques to evaluate them. In addition, Lehtola et al. (2004) identified challenges of requirement prioritisation, a very important practice in RE that may affect decision making, the development process and the rest of the development life cycle. Also, the challenges of requirement prioritisation impact on client satisfaction. Lehtola et al. conducted two case studies and, through them, present the point effect requirement prioritisation from the different perspective of customers, business and implementors, as shown in Figure 2.11. Another literature review conducted by Achimugu et al. (2014) on requirement prioritisation presented findings on the importance of prioritising software development activities and of close negotiation to make accurate decisions that enhance the quality of the process and reduce rejected requirements.

Another aspect of customisation requirements for PS products is decision making, and the impact of this practice on the software development process. Decision making is critical to the success or failure of software development projects, affecting the software development process as well as its quality, especially in dealing with internal and external clients (Brinkkemper, Ebert & Versendaal, 2006). Damian and Zowghi (2002a) investigated requirements negotiation in the GSD domain. They indicated a lack of research on requirements negotiation in the DSD domain, specifically into the challenges to software development practice such as inadequate communication, time zone and cultural boundaries.

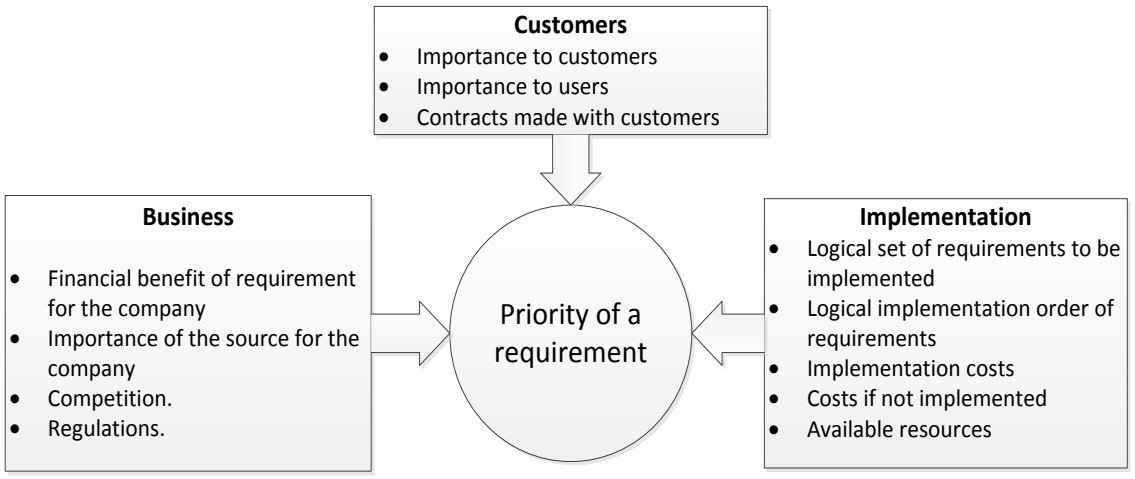


Figure 2.11: Three points of views affecting the priority of a requirement (Lehtola et al., 2004)

They concluded their research by emphasising the significant impact of requirements negotiation and prioritisation activity on delays in the development process and the positive impact of informal communication between clients and decision makers. They stated that 'The ineffectiveness of formal decision making meetings combined with the use of e-mail in resolving issues led to decisions being delayed and issues remaining unresolved longer than necessary.'

To summarise the previous section, the three software development areas of DSD, RE and locality development using an Agile approach shown in Figure 2.1 represent challenges and benefits for the software development industry. However, the overlap of those areas is referred to in the literature as a gap that needs more research into the challenges and opportunities, especially considering the communication of customisation requirements in distributed projects, as most research discussed in the RE literature focuses on bespoke products, not market-driven software products such as PS. Furthermore, much of the research has discussed the importance of decision making in the development process and its impact on different aspects of software development projects in terms of both the quality and productivity of process and product.

2.6 Chapter Summary

This chapter has presented the literature in three areas involved in the development process, as shown in Figure 2.1. The first part of this chapter discussed DSD by showing different models of development projects, classified by contract and geography. Next the research trend of DSD was presented by reviewing studies that have discussed the challenges and opportunities of distributed development projects, including solutions proposed in this context. Section 2.2

concluded with the outputs of DSD studies, representing the main challenges of DSD projects and suggestions by DSD researchers Table 2.2.

The second part of this chapter is an RE review, discussing the importance of RE in software development in general and DSD projects in particular. RE is referred to as a significant part of software development industry that has attracted the attention of both researchers and reactionaries. Thus, researchers have discussed RE practices for different software products such as bespoke and PS products, as explained in section 2.3. The challenges and opportunities of RE on distributed development projects are also reviewed such as cultural diversity, inadequate communication, knowledge management, which are also mentioned in Table 2.2.

Next, the Agile software development approach was described as it plays a significant part in transforming the software development process from a traditional operation to a lighter process using practices such as face-to-face meetings and informal communication at the locality. Consequently, software development projects and practices, including DSD and RE, benefit from Agile development techniques and local negotiation and decision making.

Section 2.4 presented a review of the literature discussing Agile phenomena, the challenges and opportunities for DSD and RE domains, including proposed solutions using Agile to solve or diminish those challenges such as challenges of Agile adoption, requirements engineering challenges with an Agile approach and locality in an Agile development approach. Finally, section 2.5 discussed the overlap of the DSD, RE and Agile areas, as presented in Figure 2.1, in order to investigate the effect of local negotiation and decision making on the customisation requirements of distributed development projects. The section presented the importance of this scope and the recommendations given from the perspective of different researchers to focus on conducting more empirical research on the challenges of customising PS products in the distributed domain and on requirements practice on this area, as most research deals with this issue from the bespoke prospective, and there is a difference between bespoke and PS products as seen in section 2.5. The following table, Table 2.2, shows the challenges and factors identified in the literature that impact on the customisation process in the DSD domain, portrayed as a framework for the issues associated with customising software in DSD (FCCSD), then explained and confirmed by 19 practitioners in the following chapter.

Table 2.2: Challenges and factors impacting on DSD customisation identified from literature review

Source	Factors
Coram and Bohner, 2005	Project management, people, planning, documentation, development process.
Jiménez et al., 2009	Cultural differences, group awareness, configuration management, knowledge management, coordination, collaboration, project and process management, process support, risk management, quality and measurement.
Sengupta et al., 2006	Collaborative software tools, knowledge acquisition and management, testing in DSD, process and metrics issues.
Damian and Zowghi, 2007	RE and its challenges in DSD, such as technology, culture and informal communication.
Fowler, 2003	Cultural changes, requirements, documentation, costs, project management and future of DSD.
Portillo-Rodríguez & Ebert, 2010	Discussed collaborative technologies such as IBM Jazz, Microsoft SharePoint, Google Apps and IBM Lotus.
Therrien, 2008	Communication, time zones, cultural challenges, trust and enablement factors and technical challenges such as communication tools.

Chapter 3: Framework for Challenges of Customising Software Products across Organisational Boundaries in DSD

3.1 Introduction

In the literature review on customising software in the distributed domain (Chapter 2), many studies were discussed and the challenges and key factors of developing software for DSD and multi-site development outlined. In addition, several proposed solutions were described, as well as models and frameworks for DSD (Schneider, Torkar & Gorschek, 2012). However, there is a lack of literature on customising software products across organisational boundaries in DSD using local practices such as the Agile approach. Therefore, one of the main objectives of this research is to explore the domain of customising software across organisational boundaries (discussed in section 2.5) and to identify the challenges of the process to inform researchers and practitioners.

This chapter identifies the challenges of customising software across organisational boundaries in the DSD context. To achieve this goal, various aspects concerning outsourcing software products in the distributed domain have been reviewed such as DSD, DAD, customisation software, organisational boundaries and RE. Miles and Huberman (1994) define the conceptual framework as one that *'explains, either graphically or in narrative form, the main things to be studied, the key factors, concepts, or variables and the presumed relationships among them'* (p. 18). Hence, a framework was designed to help researchers, practitioners and stakeholders to understand and consider the challenges of the customisation process across organisational boundaries in the distributed domain. It was used to make explicit the challenges identified in the literature, to evaluate their relevance and to identify the important components.

The following sections present the design process of this framework (section 3.2.1), termed the Framework for Challenges of Customising Software Products across Organisational Boundaries in DSD, or FCCSD, with a description of its components (section 3.2.2), followed by a preliminary study to confirm the framework contents in terms of challenges for customisation software products in the distributed domain (section 3.3). After that there is a discussion of the findings from the literature review and the preliminary study, presenting the scope of this research and indicating the gaps in the literature, as shown in section 3.4.

3.2 FCCSD Framework

The framework for the challenges for customisation software products across organisational boundaries in distributed projects (FCCSD), shown in Figure 3.1, consists of four components: communication, project management, knowledge management and configuration management. These cover management aspects as well as the software development process of projects in the distributed domain. The rest of this section presents the design process and the content of the FCCSD framework.

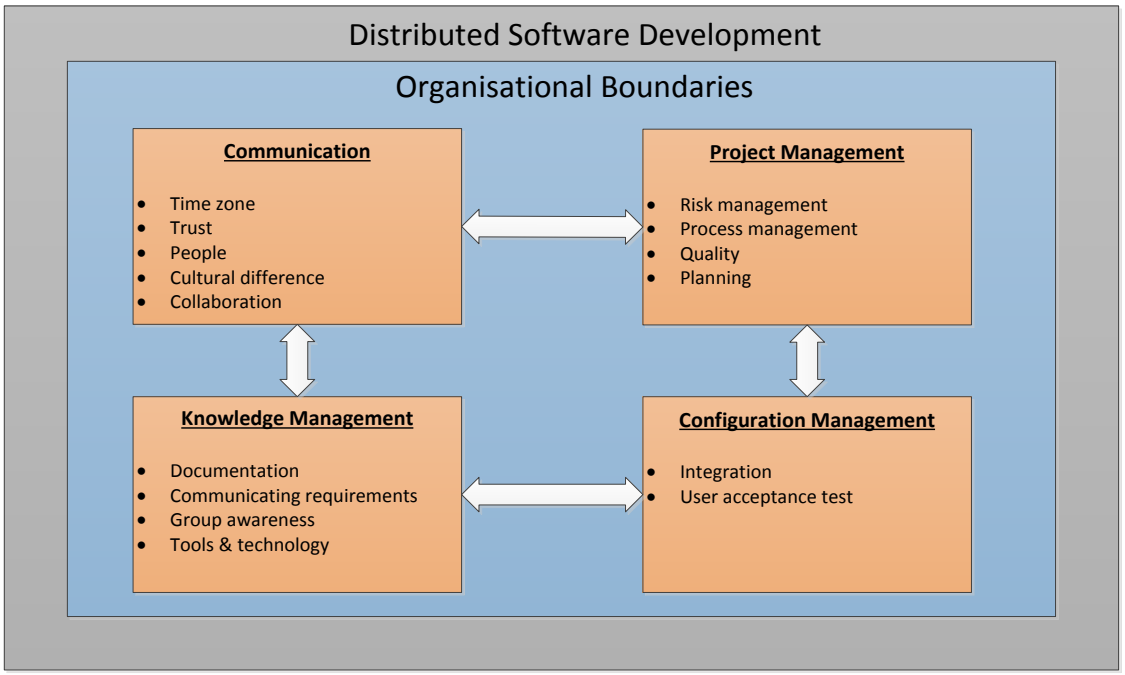


Figure 3.1: FCCSD: Framework for challenges of customisation software across organisational boundaries in distributed projects

3.2.1 Framework design process

The main goal of the FCCSD is to articulate the challenges for customisation of software across organisational boundaries for distributed development projects. The components were identified by reviewing related topics in the research domain. Figure 3.2, below, illustrates the process of designing and collecting these components, starting with a review of the challenges and key factors of related topics in the framework domain: software outsourcing; customisation software development; DSD; DAD; and organisational boundaries. The challenges of the research domain were then determined, as shown in Level 2. Finally, a survey was conducted to evaluate the identified challenges for customisation software products in the DSD domain to confirm these components, yielding the FCCSD, shown above (Figure 3.1).

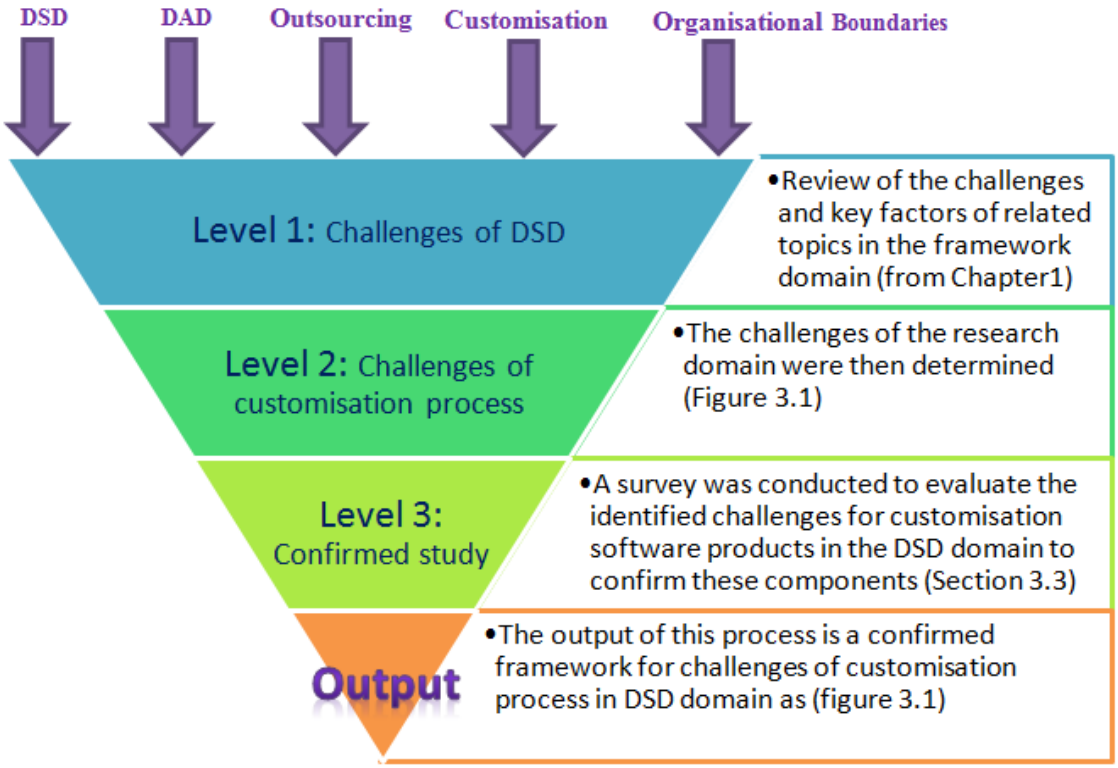


Figure 3.2: FCCSD design process

3.2.2 Framework components

The FCCSD is divided into four main sections: communication, project management, knowledge management and configuration management. Each section has a number of components. The following paragraphs describe each component, and the research conducted and discussed in the literature.

3.2.3 Communication

DSD has become a new trend in the software industry in recent years (Korkala & Abrahamsson, 2007), and communication has proved itself as one of the key factors. Many researchers have addressed communication as one of the main issues of DSD, besides Agile development (Fowler, 2003; Sengupta et al., 2006; Abrahamsson et al., 2002; Jiménez et al., 2009). The reason why it is important is that development in general requires close communication; this only increases with Agile development, which emphasises face-to-face interaction. Discussion on communication is a key factor affecting customisation of software products across organisational boundaries in the context of DSD, including related topics such as the following.

- **Time zone:** Time zone is an influential factor in communication for teams spread across countries, as well as the working hours in different organisations. Agile software

development promotes people interaction during the development process and this is difficult if there are different time zones. Many of the reviewed studies emphasise this effect on communication in the distributed domain. In addition, communication among teams in different time zones is a future research trend in DSD that needs more investigation and proposes solutions to minimise the impact on the development process (Sengupta et al., 2006).

- **Trust:** During distributed development and development across organisations, the problem of face-to-face communication highlights another issue, namely trust between team members at different stages and in different forms, such as in requirements negotiation, the exchange of information and conveying experiences. Trust has an impact on many aspects such as communication and collaboration, project management and knowledge management. Da Silva, França & Prikladnicki (2010) maintain that levels of trust influence a development project's success due to its significant impact on the progress of a team, and consequently the project. Trust plays a critical role in outsourcing projects, as any failure in creating confidence among outsourcing companies and development teams may cause serious problems (Ali Babar, Verner & Nguyen, 2007). In addition, Khan, Niazi and Ahmad (2009) argue that having skilled human resources is an important factor in building a successful relationship between client and vendor organisations.
- **People:** The manifesto for Agile software development places great emphasis on people: 'Individuals and interactions over processes and tools' (Kent Beck et al., 2001). Furthermore, the main motivation for an organisation to distribute its development projects is to look for highly skilled human resources (Khan, Niazi & Ahmad, 2011). Personnel structures in software development and the customisation process across organisational boundaries, including project managers, stakeholders and developers, are a key factor in communication, including the development team as well as users since the Agile locality concept emphasises that they should be involved as members of the development team. Also, human resources play a vital role in RE as well as in the development process in general (Hernández-López et al., 2010).
- **Cultural difference:** This is an important factor for distributing development and developing across organisations. Fowler (2003) describes cultural change as the hardest part of adopting Agile methods. Furthermore, culture can have an effect on communication, especially in GSD projects and development on multiple sites. The challenge of cultural differences is obvious in software customisation projects for different clients (Holmstrom et al., 2006). Schmid (2014) indicates that cultural diversity is a significant problem in requirements communication and it might constitute an essential difference between vendor and client organisations when each is from different cultures, as is evident in projects run across borders.

- **Collaboration:** One of the four values of the Agile Manifesto is customer collaboration (Kent Beck et al., 2001), and Agile methods duly promote the concept of collaboration with customers and other developers to support the software product and the development process. However, distributed development projects demand collaboration among distributed teams as well as clients in order to overcome the challenges of distributed development and customisation projects (Noll, Beecham & Richardson, 2010). In the customisation of software products across organisational boundaries, collaboration is the key to meeting customers' requirements and avoiding problems of distributed sites as well as applying the Agile principles in that domain across an organisation to gain the benefits of locality. These benefits are where client, developer and project manager are involved in each development process, from requirement gathering to delivering software to meet a client's needs (Hossain et al., 2009; Kiani, Šmite & Riaz, 2013).

3.2.4 Project management

The British standard (BS6079)[1] for project management defines it as the planning, monitoring and control of all aspects of a project (Atkinson, 1999). Therefore, project management is an important aspect of software development in which the development process is managed from planning through to the delivery of working software to users. From the literature review, it may be seen that project management is a key topic for researchers in terms of applying Agile principles (da Silva, França & Prikladnicki, 2010; Coram & Bohner, 2005; Lee & Yong, 2009; Hayata & Han, 2011) to distributed development, due to its effect on the development process. These researchers discuss the following different aspects of project management.

- **Risk management:** Risk management has become a critical concern for people in DSD (Mudumba & Lee, 2010). Moreover, concern has increased with the application of Agile principles to DSD projects and across organisational boundaries, making researchers discuss risk management as one of the key challenges of DSD and DAD. This framework assumes that risk management is a part of project management and it is classified as a sub-component.
- **Process management:** Process, in the proposed framework, refers to the software development process, defined as a set of activities that aid the production of software products. These activities include requirement collection and analysis, implementation, testing and delivery (Sommerville, 2011). All these processes should be considered in terms of Agile development and DSD across organisational boundaries (Krishna, Sahay &

[1] British Standard in Project Management 6079, ISBN 0 58025597 8.

Walsham, 2006). Owing to its importance in the software development process, it is addressed as a significant component of project management.

- **Quality:** Although the software development process across organisational boundaries aims to secure many of the advantages of using Agile principles, such as reducing the time and cost as well as increasing productivity, the quality of produced software products is important (Colomo-Palacios et al., 2014). It is addressed as an important sub-component of project management in the proposed framework. In addition, quality in software quality and software process is referred to as an important measure of a software development project's success in delivering software products that meet clients' needs efficiently (Bhat, Gupta & Murthy, 2006; Jebreen, 2014).
- **Planning:** Planning is important in Agile development, such as to prioritise customer requirements before an iteration (Vlaanderen et al., 2011). However, its importance increases when arranging the distributed development process and plans across organisational boundaries (Colomo-Palacios, Soto-Acosta & García-Peñalvo, 2012). Prioritisation also plays a vital role in planning in software development process to help in decision making and dealing with client requirements, which is referred as a most important factor for RE either in development or customisation process for software development projects (Achimugu et al., 2014). Thus, the framework considers the importance of planning in project management.

3.2.5 Knowledge management

During the process of development of any software project or business there is a significant amount of information as well as knowledge to be exchanged (Bowen & Maurer, 2002). The bulk of the information appears in different forms such as test cases, codes, comments and logs on source codes, project specifications and developers' and project team members' experiences and comments. Furthermore, this information needs to indicate the level of accuracy and availability through useful tools (Espinosa et al., 2007). This framework emphasises knowledge management and integration as key components of software development across distributed multiple teams.

- **Documentation:** The manifesto for Agile software development (Kent Beck et al., 2001), lays emphasis on working software rather than comprehensive documentation. However, documentation in DSD and across organisations is vital to solve the lack of face-to-face and informal communication. Motira and Herbsleb (2001) discuss documentation in GSD and stress the documentation process in DSD as a part of knowledge management.
- **Communicating requirements:** Requirements play an important role, and extensive attention is paid to them in the software development process. Much research addresses

communicating requirements as one of the main challenges in the development of distributed development projects (Damian & Zowghi, 2003; Damian, 2006). The FCCSD promotes management and software engineering practices through Agile concepts. Agile software development supports face-to-face communication and interaction with customers in the complexity of the process. The framework supports the idea of allocating Agile teams to the customer's location to gather the requirements, with other distributed Agile teams to deal with these requirements. In recent research, Schmid (2014) discusses the challenges of global RE in the GSD domain. In this study the communication of requirements across organisational boundaries is indicated as a key practice in DSD projects that may cause a significant problem if there is any difficulty. Calefato, Damian and Lanubile (2007) find requirement communication plays an important role in the development process and refer to the effectiveness of face-to-face communication to elicit and negotiate a client's requirements in a distributed situation.

- **Group awareness:** Information should be equally accessible by everyone in DAD teams, such as developers at different sites, thus group awareness is a very important factor. One of the Manifesto's values is an emphasis on individuals and interactive action (Kent Beck et al., 2001). An awareness among team members, especially in multi-teams projects and PS products customised to multiple clients, is of paramount importance to allow developers to avoid conflict with existing code or change important functions (Herbsleb, Grinter & Finholt, 2001; Fogelström, 2010).
- **Tools and technologies:** In the development process, either in traditional approaches or Agile methodology, various tools and technologies are used. These may be at the communication level or at the development and management level, such as tracking, e-mails and documentation. Much research indicates the importance of tools in the development process, especially in DSD projects, to reduce the challenges of communication (Martignoni, 2009) or to help to manage clients' requirements (Carrillo de Gea et al., 2012).

3.2.6 Configuration and integration management

The coordination and synchronisation of the source code and software versions is an important step in any iterative development. However, the integration and version control of the source code becomes more complex in distributed projects across multiple teams and organisations (del Nuevo, Piattini & Pino, 2011). Therefore, configuration management is a key component of this framework and guides developers and project managers at the customer's location to consider this step and make sure a new version is integrated with the customer's needs and environment in terms of both platform and hardware.

- **Integration:** For the process of customisation of software products across organisational boundaries, there are multiple versions to meet customers' requests for changes or new requirements (Bosch & Bosch-Sijtsema, 2010). Thus, the integration process is emphasised to make sure a new version is compatible with the current one or to customise that version for the organisational system. Also, emphasis is laid on using the version control concept and technology to work and move smoothly from version to version in the customisation process across customers' boundaries.
- **User acceptance test:** Most software testing, such as unit tests and integration tests, is performed in development time by the development team (Gopalakrishnan, 1996). However, user acceptance tests require sharing customers to make sure that the software meets all customers' requirements, thus represent one of the key components of the proposed framework to address testing across organisational boundaries. Since this research aims to identify the challenges of customising software across organisational boundaries rather than within development teams during the development life cycle or Agile development sprint, the user acceptance test has been selected as one of the challenges in that domain (Sengupta et al., 2006).

To sum up, previous sections have presented the FCCSD, describing the challenges and key factors in the process of customising software product across organisational and distributed boundaries. These sections have explained each, together with research conducted on each component. The FCCSD has four sections covering customisation, from collecting requirements to delivering and installing a solution at a client's site. These sections are communication, project management, knowledge management and configuration management. Each section has components that play an important role in the customisation process, as explained in section 3.2.

3.3 Preliminary Confirmatory Study for FCCSD

The FCCSD, introduced in previous sections, aims to describe for researchers and practitioners the challenges of customising software products across distributed and organisational boundaries. These challenges have been identified from reviewing the literature on various areas of software development relating to the customisation process, as discussed in Chapter 2, such as challenges of outsourcing, RE, DSD, DAD and development across organisational boundaries, as shown in Figure 3.2. The FCCSD intends to help researchers and practitioners, therefore this project is a preliminary study to confirm the FCCSD through researchers and practitioners in the software community. In addition, it aims to discuss the customisation of software products to confirm the importance of this area and the challenges faced with experienced researchers and practitioners in the software community. It uses a quantitative

method with an online questionnaire to collect participants' answers to specific questions regarding the research domain. The questionnaire has been selected as it is a one of the most common techniques used to collect primary data, using an identical set of questions to obtain reponses from participants (Gray, 2009). The following sections present the design of the questionnaire and describe the participants and the processes, followed by a section presenting and discussing the findings.

3.3.1 Questionnaire design

In this study the main purpose of a survey was to confirm which components of the FCCSD are the important challenges facing the customisation of software products in DSD across different organisations. The FCCSD addresses the challenges of software customisation across organisational boundaries in distributed development projects using outsourcing to customise software products, especially those that use local practices such as Agile in the distributed domain.

The questionnaire has three parts, concerning:

1- Demographic information.
2- The importance of customising software across organisational boundaries.
3- The proposed framework and its components.

According to Saunders et al. (2009), rating questions are often used to collect opinion data. Therefore, questions were devised in this format to collect participants' opinion using the following Likert-style rating scale: Strongly agree = 5; Agree = 4; Neutral =3; Disagree =2 and Strongly disagree = 1.

3.3.1.1 Ethical approval

Ethical strategy has a significant importance in an empirical study, including a survey (Gray, 2004). This survey was approved by the Ethics and Research Governance Online (ERGO) committee at the University of Southampton. The reference number for this study is 4427 (Appendix A). Although the questionnaire did not require any personal information and participants were completely anonymous, the collected data were kept confidential, used only for research purposes and deleted after the analysis of results.

3.3.1.2 Recruitment of participants

The questionnaire was administered to participants via the online software tool called SurveyGizmo2 that had been used to design the survey, and were kept anonymous. The questionnaire was widely distributed by various means: it was sent to attendees of the XP 2012 conference[3] in Sweden, the XPDay 2012 workshop[4] in London, and to more than 10 user groups on the online social networking platforms LinkedIn and Yahoo. These user groups are comprised of people with an interest in Agile and DSD.

Nineteen respondents participated in this study. Figure 3.3 illustrates the distribution of respondents from six different countries. The participants were chosen using the following criteria: 1) software researchers and practitioners including developers, system analysts, project managers and testers; and 2) experience in software development, Agile methods, customising software projects and DSD projects. The majority of participants in this study were experienced in software development, Agile methods and customisation projects (Figure 3.4).

3.3.1.3 Samples and participants

This study focused on researchers from academia and practitioners in the industry who had experience in one or more of the following:

- Agile development and Agile methods.
- DSD.
- Customisation of software projects.

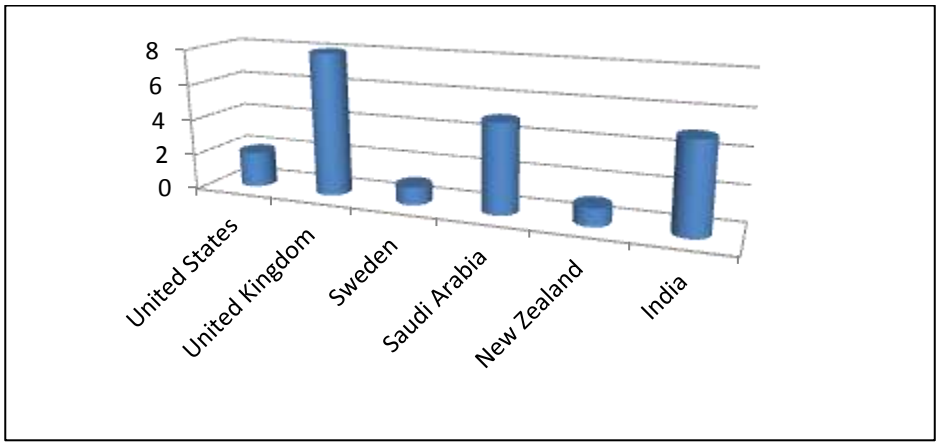


Figure 3.3: Distribution of respondents

[2] Survey Gizmo is a commercial tool for administration. It is available on http://www.surveygizmo.com/
[3] XP2012 in Malmo, Sweden, http://xp2012.org/
[4] XPDay 2012, workshop, London, http://xpday.wordpress.com/

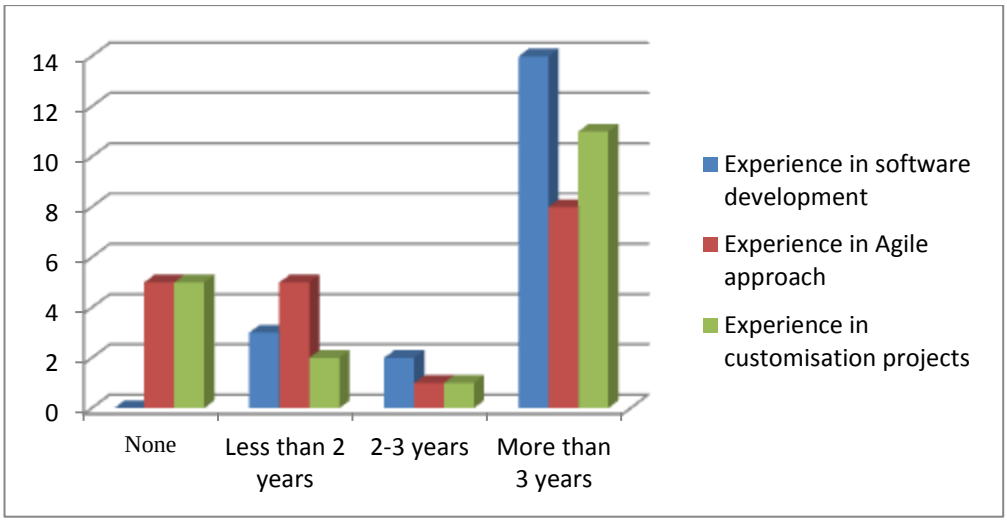


Figure 3.4: Participants' experience

To determine the sample sufficient for our study G*power software was used (Field, 2009). Figure 3.5 is a snapshot from running this software. As the main aim of this evaluation was to determine the agreement of participants on the framework components, the one-sample t test with the following settings were performed.

I. **Effect size:** 0.955. According to Prajapati et al. (2010, Table 3), the appropriate value of the effect size of one-sample t test is 0.955.
II. **Type I error:** (α-level = 0.05) was set, so long as there was no effect on the population (Field, 2009).
III. **Type II error** of this study is that β-level = 20 (i.e. 1-β error = 0.80).

Based on the G* power run, the sufficient sample size of those parameters is 11. However, 19 participants were involved, increasing the power of the study, according to Field (2009).

3.3.2 Data analysis

Since the crude data convey a simple picture with little meaning to readers, this study presents two types of analysis, descriptive and inferential, of the collected data. The reason behind using both forms of analysis is that, as Kranzler (2003) states, '*the quantitative data are described using descriptive statistics, whereas the inferential statistics are used to give the interpretation of a population by analysing the data from a smaller sample*'.

For inferential analysis, one-sample t test has been applied to analyse and recognise the significance of each question response. One-sample t is a one of several t tests used to infer a population mean when the standard deviation is unknown and the sample size is less than 30 (Argyrous, 2011). This study used the SPSS analytical tool version 20.0[5] as a statistical analysis

[5]www.ibm.com/software/uk/analytics/spss/

tool to manage and analyse the collected data. The data, already collected and analysed, were divided into three sections based on the questionnaire design:

- Demographic information.
- The research domain.
- The proposed framework.

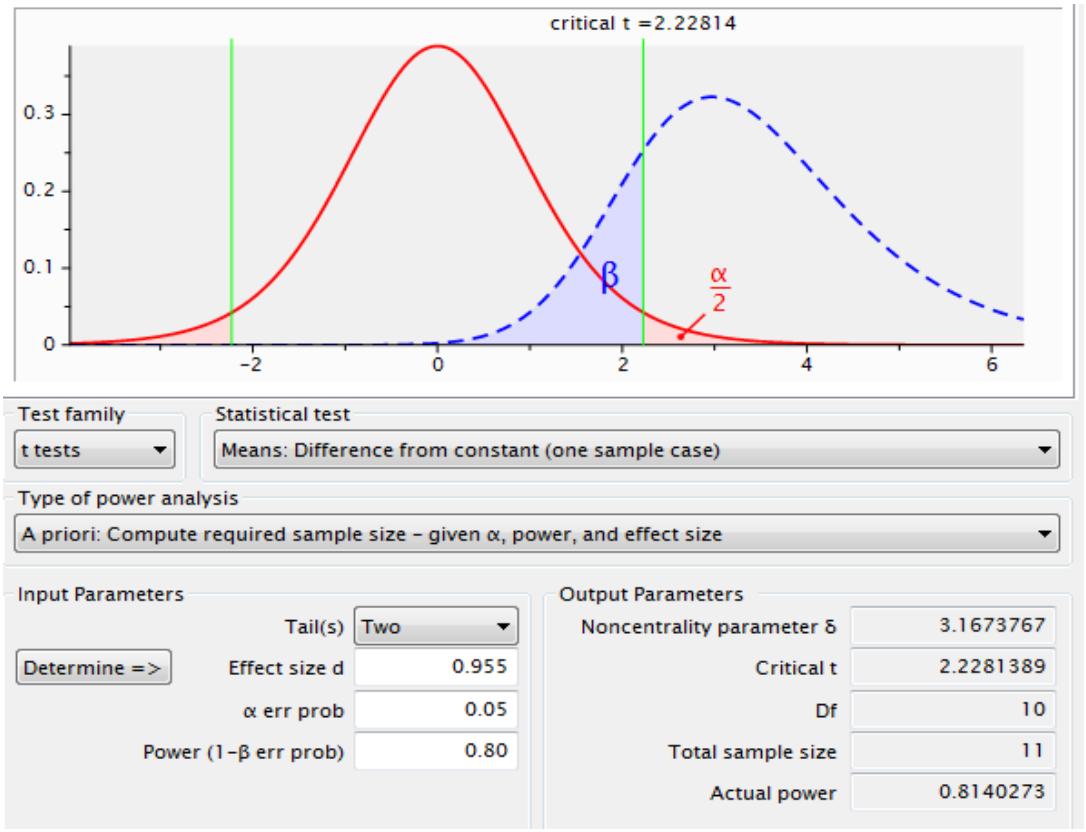


Figure 3.5: Sample size calculation by G*power software

3.3.3 Demographic information

The demographic information is required data consisting of facts about participants. Demographic data help to explain the gathered data at the analysis stage to understand the significance of the results (Fink, 2003). This questionnaire was designed to be anonymous and personal information such as name, contact number or address was not requested. Although the last question (Question 22) required contact information, it was an optional question about whether or not participants wished to be involved in a further study. This section had five categorised questions (Questions 1-5). A descriptive analysis was used to analyse and understand where the data came from. In addition, the results were presented as counts and percentages. The participants were asked questions about:

- Their position.
- Their experience in software development.
- Their experience in Agile methods.
- The type of projects they had worked on.
- Their experience in customisation projects.

3.3.3.1 Position

Figure 3.6 illustrates the positions or development roles that the respondents held. The data in this figure were computed based on the responses of 19 respondents. The pie chart in this figure contains five positions in software development and the last is 'Other', in case there were indeed other positions. In addition, the percentage for each position present is based on all the responses. It is clear from this chart there was a variety of positions. Also, two new positions have been posted: as a business analyst, and financial and banking project manager.

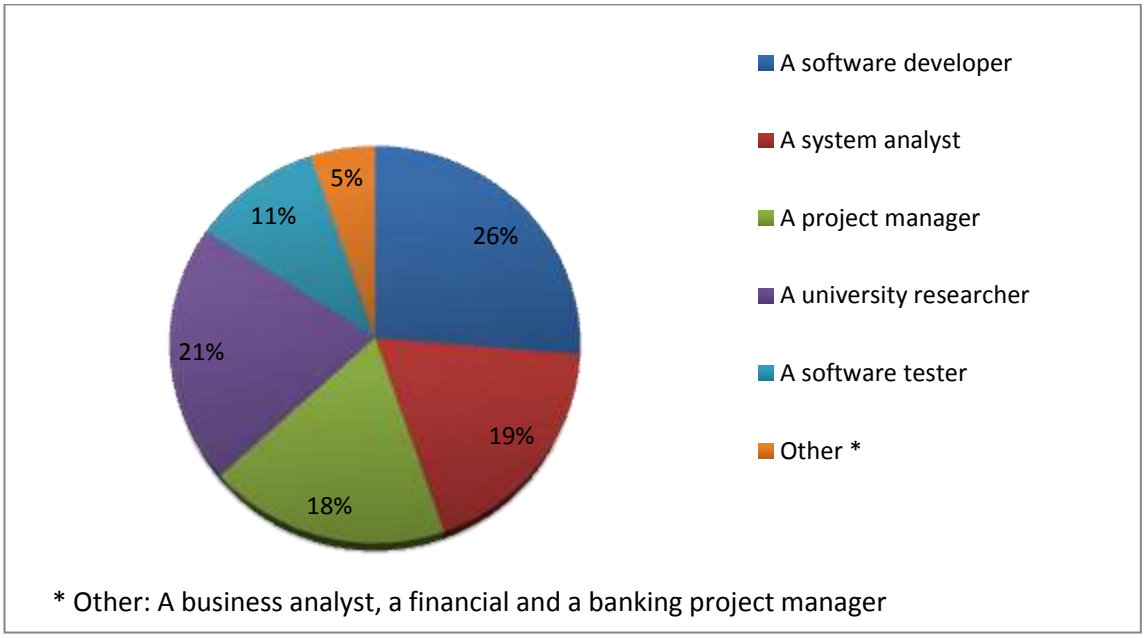


Figure 3.6: Respondents' role in this study

3.3.3.2 Experience in software development

To establish the software development experience of each participant, this question allowed a multiple choice response against four tightly defined periods of time, as in Figure 3.7. The pie chart illustrates the proportion for each period of the total responses, which was 19. Obviously, most of the expert participants' experience was of more than five years, indicating that they have mature knowledge of their role. As a result, the data reflects the reality of the research domain.

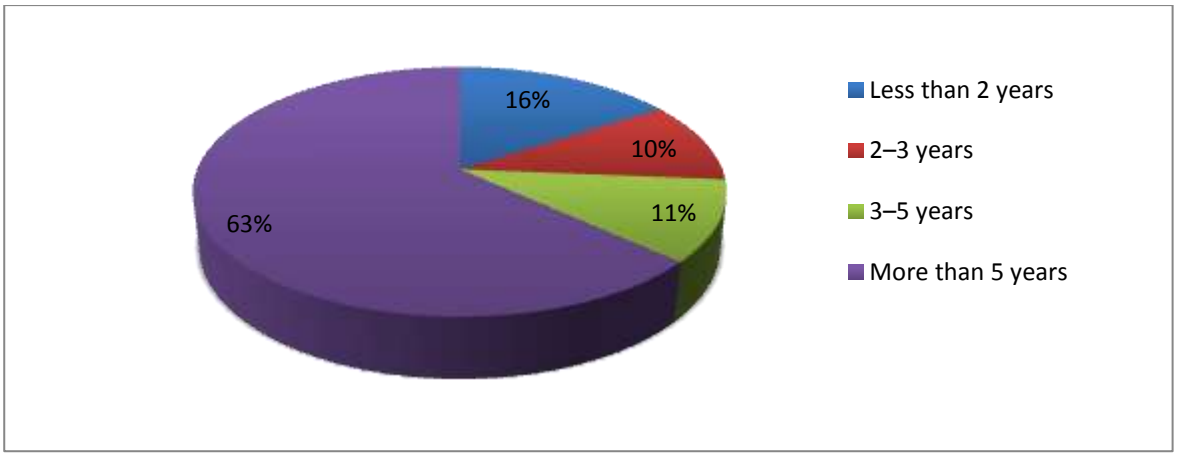


Figure 3.7: Respondents' experience in software development

3.3.3.3 Experience in Agile methods

Figure 3.8 illustrates the experience of the participants in Agile methods. The pie chart in this figure has four categories as different periods, starting with 'None', referring to people with no experience, as the questionnaire was designed to include people who had experience in software development, whether experienced in Agile or not. The chart presents the percentage in each experience category based on the total number of answers, which was 19. Some 42 per cent of the participants had experience of Agile software development, however some of the others may have had experience of other important aspects, such as DSD. In addition, the data in this figure show that the majority of participants were well-experienced in software development, Agile methods and customisation projects.

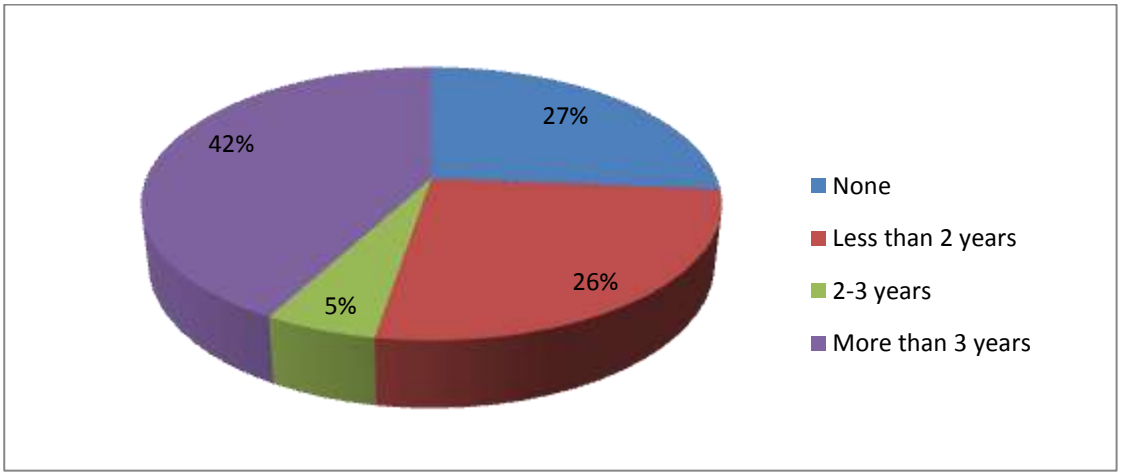


Figure 3.8: Respondents' experience in Agile methods

3.3.3.4 The type of projects that respondents had worked on

To determine the type of projects the participants had been working on, answer options were grouped into four categories:

i. Distributed projects using an Agile method (e.g. Scrum).
ii. Distributed projects using another software development approach (e.g. Waterfall model).
iii. Collocated projects using an Agile method.
iv. Collocated projects using another software development approach.

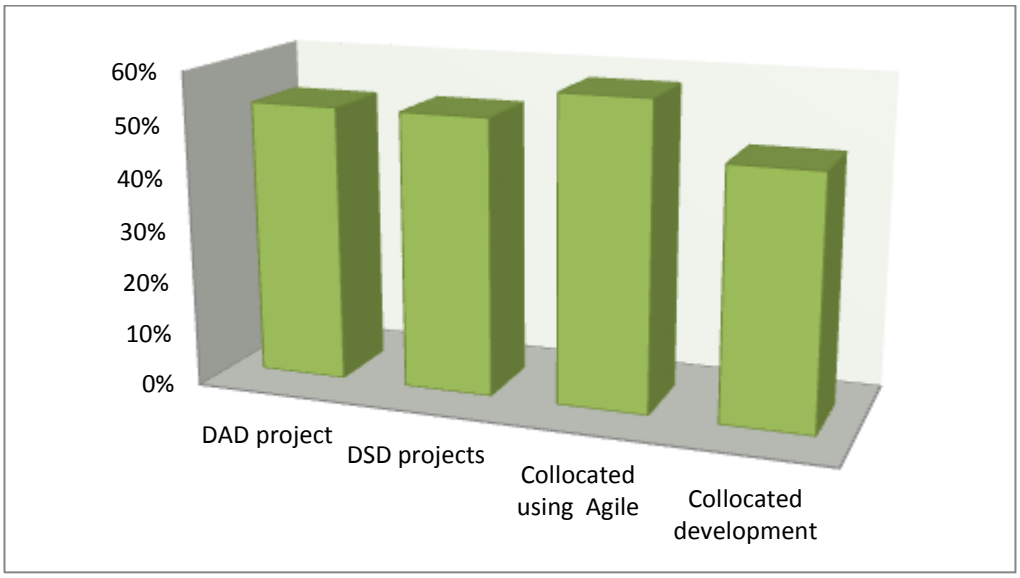


Figure 3.9: Types of projects the respondents have worked on

Figure 3.9 shows these categories. In this question participants could select more than one type, since some were involved in different types of development projects. The bar chart illustrates the percentage of the total answers. Most participants had experience in more than one type of project, for example, distributed as well as collocated projects.

3.3.3.5 Experience in customisation projects

As this questionnaire aimed to investigate the participants' opinions regarding software customising projects, Figure 3.10 reflects the participants' experience in customisation projects. The pie chart presents four categories showing the customisation project experience of participants as a percentage of responses, based on the 19 respondents.

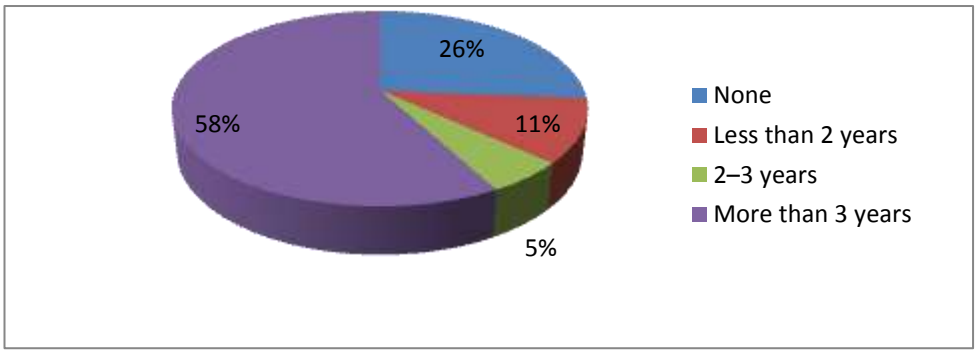


Figure 3.10: Respondents' experience in customisation projects

3.3.4 Results of research domain

This part of the questionnaire aims to determine how much customising software products in distributed domain (Figure 3.11) is interesting for researchers and practitioners.

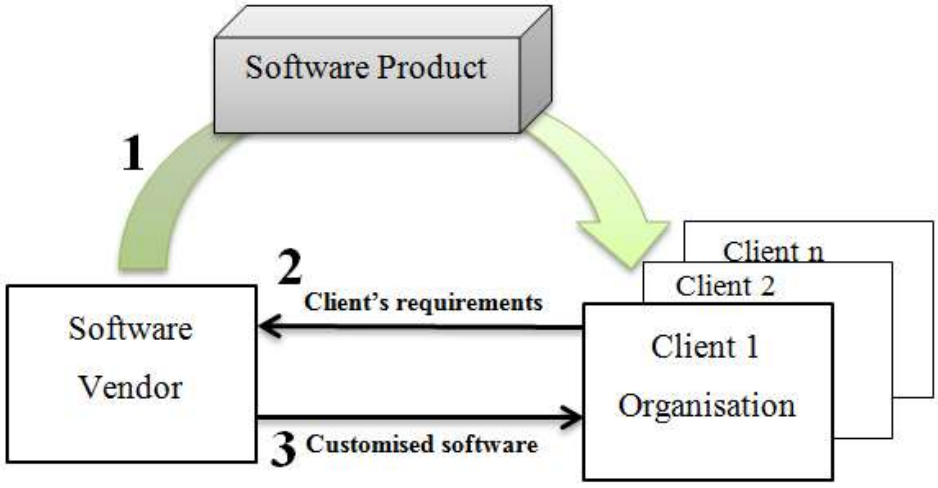


Figure 3.11: Business model between vendor and customers

Table 3.1 consists of five questions that were aimed at determining the extent to which customising software was interesting and to which participants agreed that the components in the FCCSD framework represent the challenges in the customisation process in the DSD domain. Furthermore, some of the questions discuss the current need for frameworks to identify the challenges of using Agile software development across organisational boundaries to customise software for distributed development projects.

Table 3.2 displays the result of the descriptive analysis of Questions 6–10: the means of the responses, standard deviation and standard error. The means of participants' responses on Questions 6–9 are greater than the defined test value 3, the neutral value of the Likert scale used in this questionnaire (section 3.3.1).

Table 3.1: Questions in the research domain section of the questionnaire

Number	Questions
Q6	In the industry, customisation of software code levels is common. In this case, as shown in Figure 3.11, software developers provide one software product for different customers, then customise it to customer requirements.
Q7	Agile software development has been successful over the past two decades; however, applying Agile software development to distributed development projects still suffers from challenges.
Q8	Organisational (cultural) boundaries affect the development and customisation processes between organisations in DSD projects.
Q9	Customisation processes on software codes in distributed development projects need frameworks to address the issues of using Agile methods across organisational boundaries.
Q10	Current frameworks are lacking in accommodating the challenges of applying Agile methods to distributed software projects for customisation software products across organisational cultures.

*Answer options for these questions were: Strongly Agree, Agree, Disagree or Strongly Disagree

In the same context, Table 3.3 shows the results of the inferential analysis of the same questions using one-sample t test with 95 per cent confidence interval of the difference. The results shown in column 4 represent the P value of the samples for each question. It is clear that the P values of Questions 6–9 are less than 0.05, thus are statistically significant. The P value of the last question in the same table (Q10) is 0.804, which means this result is not statistically significant. However, although the mean of answers to Question 10 is shown as 3.05, there is no statistical confidence that this result is greater than 3.0, so it is not significant, considering the standard deviation is 0.209 so the confidence interval is between 3.259 and 2.481. Question 10 was about the current frameworks and their use in addressing the challenges of customising software in the distributed development domain. This question needs awareness of the current research as well as the current framework, which might be difficult for industrial practitioners.

3.3.5 Result of framework design

This section of the questionnaire focuses on the design and the FCCSD shape. The aim was to assess how much of the framework design is understandable.

Table 3.2: Descriptive result of Questions 6–10

Question	N	Mean	Std dev	Std error mean
Q6	19	3.74	.933	.214
Q7	19	3.84	.958	.220
Q8	19	3.95	.911	.209
Q9	19	3.68	1.003	.230
Q10	19	3.05	.911	.209

Table 3.3: Result of inferential analysis of Questions 6–10

Question	t	df	Sig (2-tailed)	Mean difference
Q6	3.441	18	.003	.737
Q7	3.831	18	<0.001	.842
Q8	4.532	18	<0.001	.947
Q9	2.974	18	.008	.684
Q10	.252	18	.804	.053

Table 3.5 shows the means of responses on Questions 11–15 (Table 3.4). Although all these means are greater than the defined test value, which was 3.0 apart from the means of responses to Question 14, there is no statistical confidence for these results to be greater than 3.0, if the standard error of each question is considered as shown in Table 3.5.

Furthermore, the inferential analysis of responses to these questions (Questions 11–15) using one-sample t test analysis with a 95 per cent confidence interval of the difference (Table 3.6), shows that the P value of these questions is greater than 0.05. This means statistically there is no significance in these results. The reason might be that the different positions and thinking of the participants affected their opinion of the framework design. Also, the participants might have been focusing on the components rather than the framework design. However, the main goal of this questionnaire was to evaluate the proposed framework components by measuring how much the participants agreed with them, as will be presented in the next section.

Table 3.4 : Questions 11–15

Number	Question
Q11	The overall shape of the framework clearly refers to the challenge of using Agile methods across organisational boundaries in DSD projects
Q12	The design of the framework seems solid
Q13	The design of the framework seems coherent
Q14	The framework addresses the challenges and issues associated with using Agile software development across organisational boundaries in DSD
Q15	The FCCSD framework would help development team members in their work.

*Answer options for these questions were: 1 (weak), 2, 3, 4 or 5 (strong)

Table 3.5: Descriptive results of Questions 11–15

Question	N	Mean	Std dev	Std error mean
Q11	19	3.21	.976	.224
Q12	19	3.11	1.243	.285
Q13	19	3.32	.946	.217
Q14	19	2.84	1.344	.308
Q15	19	3.26	1.327	.304

Table 3.6: Results of inferential analysis of Questions 11–15

Question	t	df	Sig (2-tailed)	Mean difference
Q11	.940	18	.360	.211
Q12	.369	18	.716	.105
Q13	1.455	18	.163	.316
Q14	-.512	18	.615	-.158
Q15	.865	18	.399	.263

3.3.6 Result of framework components

This section is about the factors in the proposed framework. Questions 16–19 cover the four parts of the framework: communication; project management; knowledge management; and

configuration management. In addition, questions in this section were designed to see if the participants agreed that the indicators of the framework part were important.

Question 16 was ‘Communication is a challenge in DSD projects. Many aspects relating to communication affect the challenges of using Agile development in DSD. How strongly are the following factors related to communication problems: time zone, trust, people, cultural difference, collaboration?’

Table 3.7 shows the descriptive analysis of Question 16, about the communication part in the FCCSD and other indicators as challenges of customisation software using an Agile approach across organisational boundaries for distributed projects. As the Likert scale that was used in this question ranged from Strongly agree to Strongly disagree, the test value selected in this study was 3.0, the neutral value between 0 and 5. The means of responses to each indicator were compared with the test value to see whether to accept or reject the result.

Additionally, to show the significance of the results an inferential analysis was conducted with the one-sample t test using the SPSS tool at the 95 per cent level of confidence. While

Table 3.8 displays the output of the one-sample t test, all the values in column 4 (Sig (2-tailed)) are less than the defined significant level, which was 5 per cent ($P<0.05$). This means that all the results of this question are significant. The results shown in Table 3.7 and

Table 3.8 significantly refer to how communication is a challenge in customisation software using an Agile approach across organisational boundaries for distributed projects. In addition, the indicators that are listed in Question 16 are related to the communication challenge.

Table 3.7: Descriptive results of Question 16

Indicator	**Mean**	**Std dev**	**Std error mean**
Time zone	4.53	.513	.118
Trust	4.32	.820	.188
People	3.95	1.026	.235
Cultural difference	3.89	1.100	.252
Collaboration	3.84	1.167	.268

N= 19

Table 3.8: Inferential results of Question 16

Question	t	df	Sig (2-tailed)	Mean difference
Time zone	12.969	18	< 0.001	1.526
Trust	6.994	18	< 0.001	1.316
People	4.025	18	.001	.947
Cultural difference	3.545	18	.002	.895
Collaboration	3.145	18	.006	.842

Question 17 was 'These factors of risk management, process management, quality and planning related to project management bring challenges to DSD using Agile methods across organisational boundaries. In addition, they affect development processes differently across distributed development teams. How strongly are these factors related to project management challenges?'

Table 3.9 and Table 3.10 both display the analysis results of Question 17. The first (Table 3.9) contains data that describe the means of the participants' responses as well as other important information such as the standard deviation and standard error of each indicator of the project management part (Risk Management, Process Management, Quality and Planning). Furthermore, the values of the third column of this Table 3.9 refer to the fact that all are greater than 3.0, the defined test value.

The second table (Table 3.10) shows the results of the inferential analysis of the collected data by one-sample t test using the SPSS tool. This table clearly displays that all the means of the indicators in this table are significant, since all the values are less than 0.05 (i.e. $P<0.05$) which was the test value for significance. Both tables support the claim about the project management and related indicators being challenges in the framework domain.

Table 3.9: Descriptive results of Question 17

Indicator	N	Mean	Std dev	Std error mean
Risk management	19	4.11	.994	.228
Process management	19	4.11	.737	.169
Quality	19	3.84	1.425	.327
Planning	19	4.26	.733	.168

Table 3.10: Inferential results of Question 17

Question	t	df	Sig (2-tailed)	Mean difference
Risk management	4.846	18	< 0.001	1.105
Process management	6.533	18	< 0.001	1.105
Quality	2.577	18	.019	.842
Planning	7.507	18	< 0.001	1.263

Question 18 was the third question in the conducted questionnaire about the framework components: 'During the development process in distributed projects, there is a great deal of information and knowledge being transferred. There are factors which relieve the challenges of knowledge management for this type of development. How strongly are the following factors related to knowledge management challenges: documentation, communicating requirements, group awareness and tools and technology?'

As with the previous questions, descriptive as well as inferential analysis was applied to the collect data. Table 3.11 presents the descriptive information of the participants' responses to each indicator including the means and the standard error, which helps us to understand this information and note that the means of all indicators are confidently greater than 3.0. Moreover, the second table (Table 3.12) refers to the same meaning from a different angle. It presents the significance of the analysed data using one-sample t test, which is clearly less than 0.05. This led to accepting that result based on the setting mentioned at the beginning of this chapter.

Table 3.11: Descriptive results of Question 18

Indicator	N	Mean	Std dev	Std error mean
Documentation	19	4.37	.895	.205
Communicating requirements	19	4.21	.976	.224
Awareness	19	4.11	.737	.169
Tools and technology	19	3.68	1.157	.265

Table 3.12: Inferential results of Question 18

Question	t	df	Sig (2-tailed)	Mean difference
Documentation	6.664	18	<0.001	1.368
Communicating requirements	5.404	18	<0.001	1.211
Awareness	6.533	18	<0.001	1.105
Tools and technology	2.577	18	.019	.684

Question 19 was about the fourth part of the proposed framework. The aim was to get the participants' views about configuration management in the context of the research domain. The question was: 'The integration and user acceptance test represents the interaction between development teams and distributed customers in integrating and deploying the delivered software tested by customers. How strongly are these factors related to configuration management problems?'

In the same context, the second table (Table 3.14) displays that both indicators are significant when the one-sample t test has been applied to the data relating to these factors. The descriptive and inferential analysis of Question 19 refers to the strong agreement of the participants with the configuration management and both the integration and the user acceptance tests as being important factors of customisation software across organisational boundaries in distributed development projects.

Table 3.13: Descriptive results of Question 19

Indicator	N	Mean	Std dev	Std error mean
Integration	19	4.16	1.015	.233
User acceptance test	19	3.89	1.197	.275

Table 3.14: Inferential results of Question 19

Question	t	df	Sig (2-tailed)	Mean difference
Integration	4.975	18	<0.001	1.158
User acceptance test	3.258	18	.004	.895

3.3.7 Further questions

In the conducted survey, Question 20 was an optional question asking participants to identify any factors missing from the proposed framework. Although participants suggested some extra factors, most of them were already in the framework under another factor. Table 3.15 shows the participants' suggestions in the left hand column and the explanation regarding each on the right.

By contrast, Question 21 asked participants whether any factors in the proposed framework should be taken off. Participants' answers are shown in Table 3.16. The right hand columns present the location of the suggestions in this report and how many times repeated.

Table 3.15: Participants' suggestions for additional factors

Factors	**Location in report**
'Feedback and learning' P16	Group awareness (section 3.2.5)
'Integrity checks' P3	Configuration management – testing and integration (section 3.2.6)
'Project progress' P4	Project management (section 3.2.4)
'Testing rather than user acceptance test' P19	There is an explanation for why the user acceptance test was used (section 3.2.6)
'Vendor requirements' P13	Project management factor (section 3.2.4)
'Communication (e.g. e-mail)' P7	Tools and technologies (section 3.2.5)
'Emphasis on the customer involvement as a first class member of the team' P17	Communicating – people (section 3.2.3)
'Communicating progress and issues' P19	Project management (section 3.2.4)
'Coordinating channels' P3	Communication factor – collaboration (section 3.2.3)
'Vendor limitation' P13	Project management role (section 3.2.4)
'Expert persons' P10	All people included experts under the communication component in the proposed framework (section 3.2.3)
'Cost/budgets' P13	Project management factor – planning (section 3.2.4)
'Locations and physical distance' P2	Time zone (section 3.2.3)
'Management of requirements change' P12	Communication requirements (section 3.2.5)

Table 3.16: Participants' suggestions of which factors to remove

Factors	Times repeated	Location in report
Group awareness	3	This factor could be general in the framework. However, there is an explanation of this term with some references in section 3.2.5. Also, group awareness is important due to the fact that it includes customer feedback and the exchange of knowledge and information between teams in the DSD domain.
People	2	This is a key factor in software development, especially in Agile approach that includes customers, developers and project managers. Also, the Agile Manifesto (Kent Beck et al., 2001) emphasises the human factor in the development process. In addition, the description of this factor is shown in section 3.2.3.
Tools and technologies	3	Although the distributed teams in the DSD domain use various tools and technologies to simplify the communication between distributed teams (e.g. e-mail), these bring some difficulties to the customisation process in the DSD domain. Chapter 2 presents examples of these tools (Table 2.1) and the challenges of these factors in the development and customisation process in the DSD environment.
Cultural differences	1	This research focuses on the challenges of customising software products across organisational boundaries. Therefore, cultural difference is one of the main challenges to development across different organisations, sometimes located in different countries. Chapter 2 discusses this factor in detail.
Trust	1	Trust between distributed teams can become a challenge and has been discussed in the literature review in detail (section 3.2.3). Also, it plays a key role in the customisation process with distributed customers and teams.
Time zone	1	Time zone is the one of main challenges of developing software in a GSD. Chapter 2 presents the importance of time zone in DSD, as well as the customisation process.

3.3.8 Discussion of preliminary study results

The preliminary confirmatory study described in this chapter aims to confirm challenges of customisating software products across distributed domains. These challenges have been discussed in this chapter in the form of a framework (as shown in Figure 3.1). Therefore, the confirmatory study has discussed two main parts:

I. Importance of the study domain, which is the customisation of products in the distributed domain.

II. FCCSD, including its design and content.

A total of 19 participants participated in this study. They were software development practitioners and researchers involved in different software development projects, distributed, Agile and customisation process. As present in section 3.3.3, 63 per cent of participants had more than five years' experience in software development and 73 per cent were involved in Agile development projects, 52 per cent in distributed development projects. Some 74 per cent of this study's participants have been involved in customisation projects for software products, 58 per cent with more than three years' experience in this type of project. These figures indicate that experienced researchers well able to understand the challenges of this domain and assess the FCCSD were involved in this study.

The second part of this questionnaire focused on the customisation of software products across distributed and organisational boundaries as an interesting domain for researchers and practitioners of software development. The results of research referring to the customisation of software product are of interest to research and industry, areas that are in need of more focus. Also, it indicates that practitioners are faced with challenges in Agile projects, especially in DSD projects, confirming the findings from the literature that this domain still play a significant role in software development projects, as shown in the statistically significant results in Table 3.2 and Table 3.3.

The third part of the questionnaire emphasised the framework content and design. The first section discussed the design of FCCSD and how to present the challenges of the customisation process. Although the results mostly fell above the accepted level of 3.0, they were not statistically significant as opinion on design is subjective and will vary hugely between participants. However, the main goal of this study was to confirm the importance of customisation process as a domain and to confirm the identified challenges of that domain. The second section of this part discussed the challenges in FCCSD, divided into four main groups of challenges and factors of customisation software products across distributed and organisational boundaries, namely communication, project management, knowledge management and configuration management. The results of the confirmatory study on these challenges indicate clear agreement by participants on the challenges in FCCSD; all questions were statistically significant at the 95 per cent level of confidence. However, there are some challenges and factors mentioned as missing factors that need more investigation. Table 3.15 answers this

point, indicating where each factor is mentioned in the framework. Also, some factors were recommended to be removed; Table 3.16 presents these factors and the reasons.

3.4 From Literature Review and Findings to Research Scope

Chapter 2 reviewed three different domains of software development projects, namely DSD, the Agile development approach and RE, as shown in Figure 2.1. The DSD field has taken a significant interest in the software development industry, with its significant demand for outsourcing and offshore development projects to reduce costs and create high quality products (da Silva, França & Prikladnicki, 2010b). Much research has discussed the challenges and opportunities of this area, such as the communication among distributed teams, identified as one of the main challenges facing distributed development projects, as discussed in section 2.2.

Agile software development has changed the software development philosophy by reducing the amount of communication through using local negotiation among clients, developers and decision makers (Abrahamsson et al., 2002). Also, Agile development eliminates the huge effort spent on creating documents in the traditional development approach, instead adopting flexible techniques for dealing with challenges and communicating requirements in order to produce incremental software products, as discussed in section 2.4. The Agile development approach has had significant success in collocated as well as distributed projects, especially in reducing the impact of communication challenges in distributed projects (section 2.4.2).

The third software development domain discussed in Chapter 2 is RE. This is a software development practice that plays a vital role in the software development process and faces challenges from various directions, as discussed in section 2.3. Communication is a foremost concern in distributed development projects, especially relaying requirements between clients and development teams. While it is difficult to manage these requirements locally, it is even harder to communicate them over organisational boundaries and to distribute them to multiple customers (Damian, 2002).

In addition, Chapter 2 discussed the process of customising PS products, as reviewed in section 2.3.1. Much research has indicated the importance of this process for PS products in the industry, especially in recent years with the significant growth in demand for this type of software (section 2.3.1). Various aspects were discussed such as distributed development and RE, especially RE of bespoke software products compared to PS products, as investigated and discussed in much research. Recent research indicates a lack of studies that focus on RE practices of PS products, including the processes and tools of this domain (section 2.3.1).

The overlap between the customisation of software products, distributed development and Agile development domains, traced in order to discuss the RE challenges, has identified an interesting area without much research literature. Many studies discuss the challenges and opportunities of RE with DSD, or RE with Agile with DSD projects. However, the process of customisation of software products in the distributed domain using a lighter development approach has not been investigated as much as bespoke software products, as mentioned in section 3.3.1. Thus, the scope of this research lies in the customisation of software products in the distributed domain.

The challenges of this domain have been identified in the literature, as shown in Table 2.2 in Chapter 2. These challenges are discussed in the FCCSD, confirmed in this chapter by a survey of 19 researchers and practitioners from the software development community. However, in terms of the importance of communicating of customisation requirements, as indicated in section 2.3 this research is designed to focus on the challenges for multiple clients in the distributed domain. Its scope is discussing the impact of making decisions and development of clients' requirements at their location, with all the implications for communication across distributed boundaries.

3.5 Chapter Summary

This chapter presented the FCCSD (see Figure 3.1), which identified the challenges to the process of customising software products across organisational boundaries in distributed development projects, as discussed in Chapter 2. It also presented the process design, as shown in Figure 3.2, then discussed its components in details. A preliminary confirmatory study was conducted in the form of a questionnaire with 19 researchers and practitioners from the software development community in order to confirm the importance of the process of customising software products as a domain and also its challenges, presented though the FCCSD.

The main purpose of a survey was to confirm which components of the FCCSD are the important challenges facing the customisation of software products in DSD across different organisations, therefore the results indicate the clear agreement of participants on the challenges in FCCSD; all questions were statistically significant at the 95 per cent level of confidence. However, there are some challenges and factors mentioned as missing factors that need more investigation. Table 3.15 answers this point, indicating where each factor is mentioned in the framework. Also, some factors were recommended for removal; Table 3.16 presents these factors and the reasons, as discussed in section 3.3.8. Next, the research gap and scope of this research were discussed in section 3.4, indicating the importance of customisation software products in the distributed domain. Thus, this study's scope is to discuss communicating

customisation requirements in the distributed domain by exploring the impact of making decisions and development on clients' requirements at their location, and the implications for communication across distributed boundaries.

Chapter 4: A Model for Communicating Customisation Requirements of Multi-Clients in Distributed Domain

4.1 Introduction

The software industry has shifted its attention to distributed development and outsourcing (Damian et al., 2004). Previous chapters have discussed the software customisation process and challenges faced in the distributed domain, as in section 3.3.8. To meet the challenges involved, changes in software engineering practices are needed. RE plays a vital role in the software development process, as discussed in section 2.3, so managing client requirements is a principal factor in product development. While it is difficult to meet these requirements locally, it is even more difficult to communicate them across organisational boundaries and multiple clients (Damian, 2002). This level of difficulty increases in DSD projects as well as projects that have multiple stakeholders in different organisations. Since the 1990s a significant shift from co-located forms of development to GSD has taken place, demanding a high level of collaboration in the distributed domain (Damian and Zowghi, 2003; Damian, 2007). RE in a distributed setting is a complex intersection phenomenon with technical, social and organisational aspects (Damian, 2007).

In previous chapters, Chapter 2 has referred to the research conducted in DSD and its challenges, while Chapter 3 has presented the framework for challenges of the customisation process with multi-clients. These chapters emphasised the importance of communication in distributed projects and how its challenges have a major impact on other aspects of customisation for the client, such as delays in decision making, in the development process and for the entire customisation. Accordingly, this research is designed to focus on the challenge of communicating customisation requirements for multi-clients in the distributed domain. Its scope is the discussion of the impact of making decisions and development upon clients' requirements, at their location, and the implications for communication across distributed boundaries.

This chapter presents a model for communicating the customisation requirements of multi-clients in a distributed domain (CCRD). The CCRD model has two scenarios: the first locates the decisions for the client's requirements, and the second the decision-making and development process. Both aim to reduce the impact of communicating customisation requirements in the distributed domain. The rest of this chapter discusses the importance of communicating requirements in the software development process and the challenges of

distributed projects, especially for multiple clients across organisational boundaries. Furthermore, it discusses the significance of decision making in the software development process as well as the customisation process, and confirms the benefits of local decision making and development recommended by some development projects or specific approaches like Agile method, as the CCRD model is based on the locality of decision making and development in terms of reducing communication challenges in the defined domain. The final sections describe the CCRD model in both scenarios.

4.2 Communicating Customisation Requirements in the Distributed Domain

Communication is one of the most important factors in the software development process because development in general requires close communication throughout, starting by approaching clients to decide their requirements and then discussing various processes during the cycle (Fowler, 2003; Sengupta et al., 2006; Jiménez et al., 2009). This significant role of communication in software development is enhanced in a distributed domain, where the actors in the development projects are physical and need to overcome the challenges of distance communication. According to Abrahamsson et al. (2002), many researchers regard communication as one of the main issues in DSD.

On the other hand, clients' requirements are central to the software development process and are affected by the communication challenges, during the past two decades giving rise to the fields of RE and requirements management. The growing amount of research and publications shows great interest in academia and industry (Jiao & Chen, 2006). Many aspects of both distributed global development and distributed development environments in general have been discussed either to identify challenges of RE in DSD domains or to propose solutions for issues in that domain such as the communication, collection and analysis of clients' requirements (Damian & Zowghi, 2003). Moreover, customised software has become commonplace in the software industry due to the boom in outsourcing software and the offshore development process; such products are now distributed over organisational boundaries and multi-site organisations (Damian & Zowghi, 2002b).

Most existing research has focused on development software and the development management of either co-located or distributed projects, with little attention being paid to the requirements of customised software and its development process. The process of customising software and customisation requirements in DSD environments needs to be investigated in order to understand the challenges in the context of distributed software development projects (Qahtani, Wills & Gravell, 2012). The challenges take various forms such as delays in response and

misunderstandings of clients' requirements (Damian & Zowghi, 2003), and are exacerbated by having multiple clients in different geographical locations. Besides these communication challenges are engineering challenges in the distributed domain, affected again by communication issues.

4.3 Local Decision Making in the Distributed Development Domain

Research on RE for distributed domains and multiple sites emphasises that their major problems lie in communication and coordination, requirements gathering, analysis and the negotiation of requirements (Gopal et al., 2011). Damian & Zowghi (2003) consider the impact of distributed stakeholders and clients; RE activities include understanding, negotiating clients' reuqirements and making decisions on their requirements. Furthermore, distance communication on clients' requirements across multiple sites impinges on other activities such as development and project management. According to Herbsleb, Grinter and Finholt (2001), delay is one of the main problems of communication and coordination across multi-site development. In addition, there is a difference in the amount of communication between collocated and distributed teams, surprisingly smaller in a collocated workforce, regarding communication on the customisation requirements of distributed clients. Research indicates success in some cases with local decision making on clients' requirements and development, such as negotiation of clients' requirements in Agile approaches (Paetsch, Eberlein & Maurer, 2003; Moe & Aurum, 2008). In distributed projects the difficulties in what and how to communicate effectively with clients may lead to undetected conflict in assumptions and interpretations, representing a highly significant problem for distributed projects across clients' organisations, manifested in delays in decision making leading to delays in development and delivery time (Herbsleb, Grinter & Finholt, 2001).

This research aims to discuss how locating some software development practices (decision making and some development process) at clients' locations may reduce one of the clearest implications of communicating customisation requirements across distributed clients: delays in decision making on clients' requirements, the development process and the entire customisation.

4.4 The CCRD Model

The CCRD model is designed to represent the communication needed for distributed clients' customisation. It aims to enhance local decision making and development processes in order to overcome the challenges of communicating clients' requirements across multiple sites, basing the customisation process on typical practices in the software development life cycle, starting

with collecting clients' requirements through to resolving and delivering them. It relies on theories that emphasise the benefits of local negotiation and decision making, the effects of distributed development on productivity and the speed of the software development process in distributed domain (Gopal et al., 2011). Through the CCRD model, this research has investigated the benefits of taking decisions on customisation requirements and carrying out some development processes at the clients' locations in terms of reducing the implication of communicating customisation requirements challenges for multiple distributed clients.

The CCRD model considers the following two scenarios: the first is the CCRD model for local decision making. This presents the communicating customisation requirements for multiple clients using local decision making, as shown in Figure 4.1. The second is the CCRD model for local decisions and development for various clients' requirements, as shown in Figure 4.2.

4.4.1 Scenario I: CCRD-local decision making model

This first scenario of the CCRD model aims to provide a mechanism for managing and communicating customisation requirements for multi-clients across distributed boundaries. It shows the importance of communication and coordination in dealing with clients' customisation requirements for the main three groups in software development projects in the distributed domain, namely the client, the onsite customisation team (at the client's location) and the offsite customisation team (at the vendor's location) (Espinosa et al., 2007). This model (Figure 4.1) enhances the practice of local decision making at clients' locations in order to overcome the challenge of communicating and coordinating their requirements in the distributed domain and to reduce the implications of these challenges.

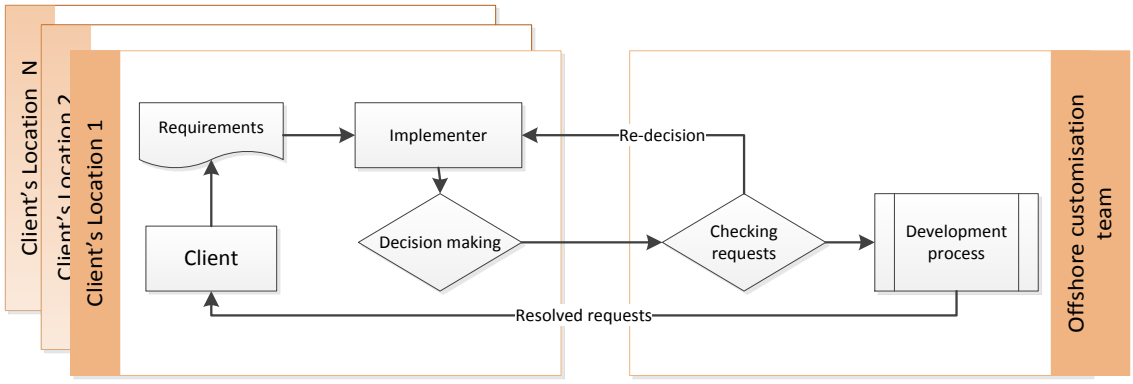


Figure 4.1: CCRD model – local decision making

As mentioned in the beginning of section 4.4, the design of this model relies on typical software development life cycle processes and practices, such as requirement gathering and analysis, design, implementation and testing (Sommerville, 2001). The benefits and features of the co-

located team and a successful experience of Agile software development rest in undertaking local negotiation and decision making at the client's location (Sureshchandra & Shrinivasavadhani, 2008). Thus, the CCRD model components start with the requirements analysis process. In this model, the collection of customisation requirements takes place at the client's location then development, including the design, the implementation and testing and evaluating processes, takes place at the software vendor's offsite location. All these components are included in the model as follows.

Customisation processes start with collecting clients' requirements. The customisation requirements in this case may be of two types: bugs, which represent problems with the working system. This type of requirement needs investigation to establish the issues and to resolve them. The second type is a new feature: clients request new features needed in the system that in most cases need analysis and development. Both types require communications between the client and the software vendor to understand the requirements and to record them. In typical offshore projects there is a small vendor team located at the client's location to carry out these activities and to act as a proxy for the main software vendor (Gopalakrishnan, 1996). The second step is taking a decision and negotiation on the recorded request, and takes three forms:

- **Cancellation:** this form of decision is made when clients change their mind about the request before a decision or after discussion.
- **Rejection:** this decision is taken after investigation finds the request is out of the project's scope, would have further implications or does not make sense.
- **Acceptance:** acceptance decisions are taken when the initial investigation finds that the request makes sense, either for bugs or new features, sending it to the central customisation team to apply the required development or to debug the issue.

The request goes through a check point with the offsite customisation team at the vendor's location to make sure the request is understood and that it is scheduled into the team's tasks. If the bug does not materialise or is unclear regarding new features, the request returns to the client's location for more investigation and a re-decision. The final process in the customisation life cycle that requests undergo is the development process, including implementation, testing and verifying, and delivery to the client's location. It is worth noting that the offsite team at the vendor's location deals with multiple distributed clients at the same time, which makes communication and coordination challenging. However, most communication in this type of project takes place at the decision making and negotiating level; in this model, decision making and negotiation moves to the client's location to reduce the challenge.

In summary, communication constitutes one of the main challenges to distributed development projects, and customisation for multiple clients struggles with same challenge. Sharing a locality

with software development and management processes has achieved success in several development approaches and contexts. This local decision-making scenario adopts the concept for customisation for multi-clients to provide a model for communicating the requirements to different clients and to overcome the challenges of the distributed domains. Therefore, this model aims to meet one of this study's hypotheses.

> *Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients.*

> *Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients.*

4.4.2 Scenario II: CCRD – local decision and some development

This scenario of CCRD is largely similar to the previous (section 4.4.1). However, the development process in the model shown in Figure 4.2 is split between onsite development practice at the client's location and offsite development practice at the vendor's location to support the distributed development practice. It aims to provide a mechanism for managing and communicating customisation requirements for multiple clients across distributed boundaries, and to enhance the concept of applying onsite development processes that have succeeded in some software development approaches, such as Agile (Fowler, 2003), recommended by some researchers as a solution to the challenges of distributed development (Gopal et al., 2011). Beside the benefits and features discussed in previous sections regarding local decision making, this scenario employs onsite decision making and development together in a single model to investigate the impact of local decisions and development on the customisation process in a distributed domain.

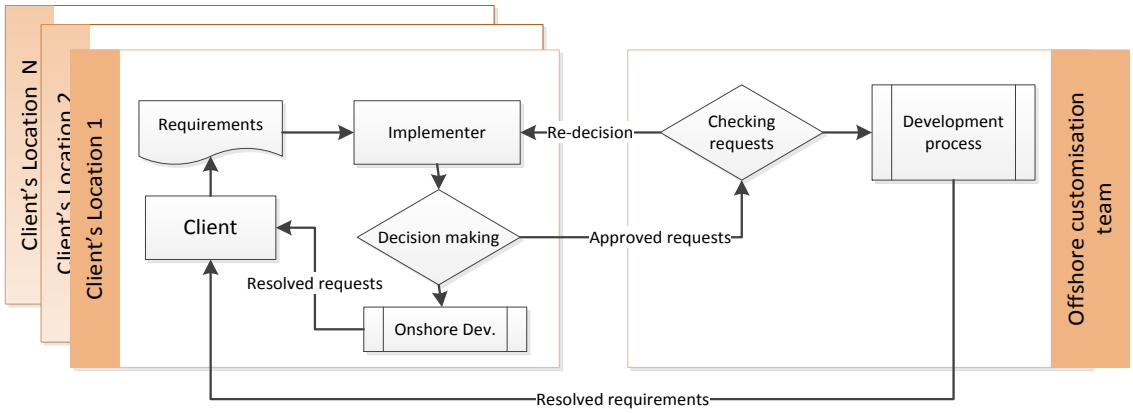


Figure 4.2 : CCRD model – local decisions and development

CCRD model

The design of this model, as mentioned in beginning of section 4.4, stems from the theories of the software development life cycle and local negotiation, decision making and development processes. Thus, its CCRD model components are from the requirements analysis process taking place at the level of the customisation at the client's location. Although the development process assumes that design, implementation, testing and evaluation process are undertaken by the offsite team at the software vendor's premises, some development processes are completed onsite at the client's location. All these components come together in the designed model (Figure 4.2), as follows.

As before, the flow of customisation processes starts with recording the clients' requirements. The customisation requirements in this case may be of two types: bugs, which represent problems with the working system. This type of requirement needs investigation to establish the issues and to debug them. The second type is a new feature: clients request new features needed in the system that in most cases need analysis and development. Both types require communications between client and software vendor to understand the requirements and to record them. In typical offshore projects there is a small vendor team located at the client's location to carry out such activities and act as a proxy for the main software vendor (Gopalakrishnan, 1996). The second step is taking a decision and negotiating the recorded request, and takes three forms, as mentioned in the previous scenario: cancellation; rejection; or acceptance.

After that, the request is allocated to the onsite team to complete the request and to deliver it to the client, or is sent to the offsite team located at vendor's location. The request goes through a check point with the offsite customisation team at the vendor's location to make sure the request is understood and that it is scheduled into the team's tasks. If the bug does not materialise or the request is unclear regarding the new features it returns to the client's location for more investigation and a re-decision. The final process in the customisation life cycle that requests undergo is the development process, including implementation, testing and verifying, and delivery to the client's location.

In this second scenario of the CCRD model (Figure 4.2), decision making is undertaken with an onsite development team at the client's location in order to invest in the features of close communication and negotiation at a client's location and to develop small or quick requirements locally. This aims to reduce the negative implications of a distributed customisation process such as development and decision times. In this regard, the following hypothesis is formulated for the impact of CCRD model for local decision making and local development:

Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process.

4.5 Chapter Summary

This chapter has described the importance of communication of customisation requirements for multiple clients across distributed boundaries (section 4.2). Also, the benefits and features of local decision making and development process have been discussed, as the concept has been used successfully in projects and approaches to software development such as the Agile method such as reducing delay time and reduce misunderstanding requirements among involved teams in customisation projects (section 4.3). This has been the motivation to discuss the impact of local decision making and development in the customisation process to reduce the challenge of communicating across distributed boundaries. Furthermore, this chapter has provided a model for the customisation process in the distributed domain. This CCRD model has the two scenarios presented in section 4.4.1 and 4.4.2, one using local decision making at the distributed clients' location, and the second enhancing the application of the development process for some clients' requirements at their location. A description of both has been presented in this chapter.

Chapter 5: Evaluating local decision making and development at client's location in DSD

5.1 Introduction

In previous chapters, the focus was on literature relating to customisation for multiple distributed clients, the motivation for this research and the proposed model for communicating customisation requirements across the distributed domain (the CCRD model). This chapter focuses on the methods and approaches used to answer the research questions (below) and to examine the following hypotheses.

RQ2: What is the impact of making decisions locally on the total decision period and the total customisation period of software customisation processes in the distributed domain?

RQ3: How much would undertaking the process of development of minor tasks at clients' locations reduce the total development and customisation period?

Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients.

Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients.

Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process.

This chapter contains several sections as illustrated in Figure 5.1. Section 5.2 presents a brief explanation of the methodology in general and also the methodology in software engineering and software development. Section 5.3 presents the contextual review study including the selection of the case study in this research, the process and its outputs. Section 5.4 presents the simulation method used in this research including the reasons to use this method, the types of simulation and the process used to fulfil the objectives of this research. This gives the setting and results of the simulation of the baseline model for the customisation process used for the selected case study, and presents the statistical tests used in the evaluation experiments to examine the defined hypothesis. After that, section 5.5 presents the confirmatory study, conducted to confirm the outputs of both the contextual review and the simulations. That section

includes a brief summary of the importance of confirmatory study in this research, the process and approaches used and their outcomes. Finally, section 5.6 summarises the whole chapter, explaining the main process and the results obtained. Although this chapter gives some results of the validation of the baseline model and its setting, the full results of each experiment are presented in Chapter 6, which aims to present the findings together with analysis and interpretation.

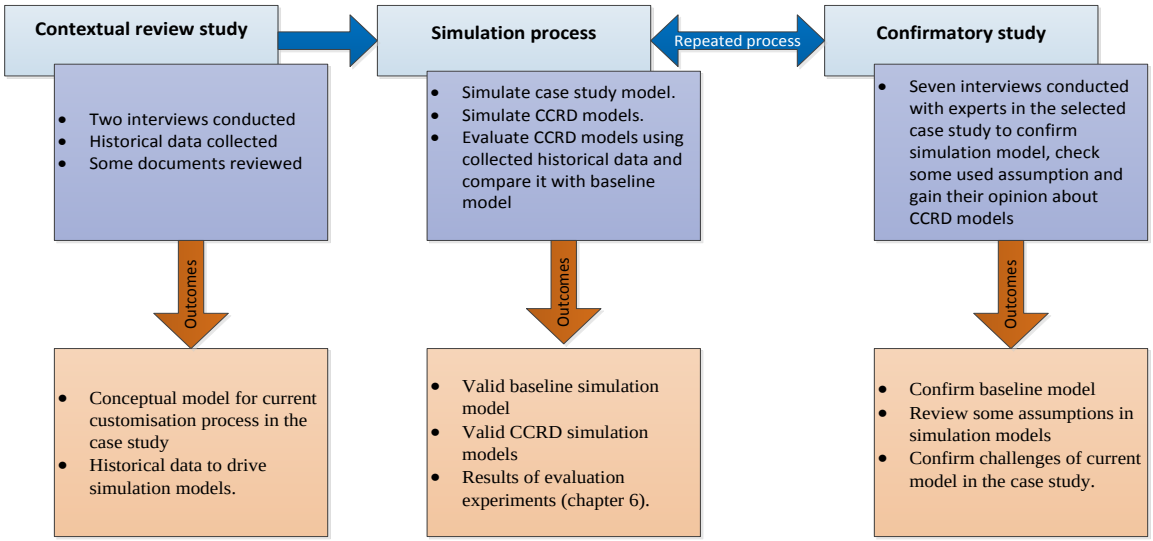


Figure 5.1: Conducted research methods and stages

5.2 Research Methodology

Research methods involve various forms of data collection, analysis and interpretation of research studies and findings (Creswell, 2008). Also, Evans, Gruba and Zobel (2011) point out that a research method is the way to select a hypothesis, answer research questions or evaluate innovations. There are many methods used for different aspects of software engineering and software development research, for example simulation, empirical research, case study and mathematical proof (Glass, Vessey & Ramesh, 2002). This research has selected simulation and empirical methods using qualitative approaches through a selected case study, as explained in the following sections.

Software development and its life cycle have many human factors, difficult to establish using quantitative measures (Hove & Anda, 2005). In addition, software engineering is a contemporary phenomenon in a natural context (Runeson & Höst, 2008). For this reason, the first method used in this research was a contextual review as a case study had been selected, involving customising software for multiple distributed clients (as explained in section 5.3). The

case study was selected as Stake (2000) states that case studies add valuable understanding and extend our experience of a subject. Also, the case study methodology was indicated as a suitable methodology for software engineering research as it investigates different phenomena in its natural context (Runeson & Höst, 2008). The semi-structured interview approach was applied in order to collect as much as data as possible on that case, and was recommended as a suitable method to collected deeper knowledge from a specific domain (Creswell, 2008). The qualitative method was applied twice in the evaluation of this research, in the contextual review (section 5.3) and the confirmatory study (section 5.5).

The software development field has increasingly used modelling and simulation to support the process of software development and software project management (Kellner et al., 1999). In addition, Kellner et al. (1999) state that the software process simulation is currently being used for different issues such as the strategic development of software development, process improvement, and control and operational, management of software engineering. In this research, the simulation approach is used for modelling. It simulates the customisation model of the selected case study and is used as a baseline model in the evaluation experiments for CCRD model in order to test the defined hypothesis of this research.

5.3 Contextual Review Study

In the mid-1980s contextual design was developed to help system designers to produce more useable systems by tapping into user experience. It adopts the concept of client-centred design that enhances the user's experience and involves him or her in the design process (Beyer & Holtzblatt, 1998). Beyer and Holtzblatt (1998) state that 'the contextual design is the backbone for organising a client-centred design process'.

Contextual inquiry is one of the techniques of contextual design used to investigate work practices in the workplace. It developed as a data collection tool using ethnographic techniques, by conducting and observing studies and interviews with users while they work (Beyer & Holtzblatt, 1998). The main goal of this process is to gain reliable knowledge about what people do in real world cases, what they care about, and the structure of the work practice including the procedures and process that they follow in order to understand the users and their work practices. Holtzblatt, Wendell and Wood (2005) refer to the interview as a powerful tool of contextual inquiry to draw out the knowledge from users in the workplace. Also, they define contextual interviews as one-on-one sessions conducted in the workplace that focus and observe on users' work and discuss their tasks.

This research used the contextual interview technique in order to understand the working of a selected real world case and to collect wider information about the company in question, such as the customisation processes, the team's structure and any information that might aid the researcher to design and build the conceptual model of current workflows in customisation at the case. This model will act as a baseline in the evaluation process of this research.

5.3.1 The real world case study

There are different ways to collect data for either building or confirming a theory. For example, the survey technique is used where large volumes of data need to be collected. By contrast, a case study explores more information and goes deeper with a focus on a range of people, organisations or contexts (Gray, 2009). According to Stake (2000), case studies add valuable understanding and extend our experience of a subject. In addition, case study methodology is indicated as suitable for software engineering research as it studies different phenomena in their natural context (Runeson & Höst, 2008). Therefore, this research used a real case study of customisation process in order to build and validate a simulation model for that case and then used it as a baseline in its evaluation.

For this reason, the company chosen was involved in customisation software and dealt with clients' requirements in the distributed domain. The company is in Saudi Arabia[6] (see map in Figure 5.2) and has significant experience in customisation products for distributed clients. The company has developed a variety of software solutions for different sectors such as academic, healthcare and business intelligence, developing and customising products to clients' needs. Furthermore, the company has established procedures for managing clients' requirements through the customising, decision-making and development processes that help this research to understand the company's current work and to design a conceptual model and a simulation model as a baseline for its evaluation.

5.3.1.1 Geographical spread for the selected company

The company selected for the contextual interviews study is a multinational company that covers a wide geographical area, as shown in Figure 5.2, and serves more than fifty clients in various countries. It has two main teams for development and customisation. The first team is located in Jordan (Figure 5.2), serving all clients. The second is in Saudi Arabia, as this country is its main client. It has more than 18 clients in the academic sector alone. As the Saudi clients mostly have the same culture and systems, the company has established its team there to support

[6] Saudi Arabia: Is the kingdom of Saudi Arabia, the largest state in western Asia http://en.wikipedia.org/wiki/Saudi_Arabia

them. Conducting this study with the company's team in Saudi Arabia reduced the effect of some variables in the distributed domain on the experiment setting, such as the time of day and cultural differences as all academic clients in Saudi Arabia have the same policies (from the Higher Education Minister) and share a time zone. However, the Saudi Arabia team follows the same process and procedures of customisation software for distributed clients as the Jordan team.

5.3.2 Interview design

The contextual interview was designed to collect information to understand the customisation process in the company and how it collated and dealt with clients' requirements in the distributed domain. Also, the contextual enquiry was used to understand the structure of the development team and the decision-making process at the company in order to obtain sufficient information to design the conceptual model of customisation at the company for the simulation model. The overview for the questions is shown in Table 5.1. The questions were designed to cover the entire process of customisation at the software and, in addition, any information related to the project's scope such as the team structure and challenges faced in the work, and the resources used in daily work.

Figure 5.2: Geographical map of company clients:
Saudi Arabia team Jordan team (from company website)

5.3.2.1 Process

At the beginning of each interview, the nature of the study and the purpose of the research were explained to the participants. A consent form was provided for them to sign to say they

understood the roles and policies of the interview as well as agreed to participate in the study. After that, the interview started by recording the exchange and taking notes to focus and interact with the participant. The voice recorder helped the interviewer to focus on the session (Hove & Anda, 2005). The recordings were transferred and then deleted. Furthermore, as this study aims to understand the real work and the current process of the company, the participants were shown some process flowcharts, examples of some requirements and the processes for those requirements, and the tools and systems used to manage their work.

5.3.2.2 Analysis and using of the collected data

Qualitative research outputs are in various forms such as interview data, scanned documents and field notes and comprise different types of data depending on the sources of information (Denzin & Lincoln, 2005). The findings of this study's collected data have been used to construct the conceptual model and simulation of the actual working processes at the company. The analysis of the interviews used notes and quotes to support the assumptions of the simulation design. While interview participants in general may provide historical data on their activities (Creswell, 2008), in this study they contribute historical data on the customisation process for 18 distributed clients, comprising 2479 requirements taking 1290 working hours. This data was collected in form of log files or numbers used as a trace data to run the simulation.

Table 5.1 Overview of interview questions

No	Questions
1	How are your company's teams structured?
2	What processes do the development, customisation, and support services follow?
3	How are requirements issued at client locations?
4	How are clients' requirements collected in distributed domains?
5	How do you communicate with your clients and your representatives at client locations?
6	How do you make decisions about clients' requirements?
7	What is the customisation process from the time an issue is created until it is closed?
8	What challenges do you face in terms of communicating clients' requirements in distributed domains?
9	Do you have additional information you wish to provide about the current customisation process or about any other issue?

5.3.3 Study outcomes

This study has two main outputs; the first is information about the customisation of multiple distributed clients' requirements in order to build a conceptual model of the selected case study, and the second is historical data of clients' requirements and the process applied to analyse it and use it in the case study simulation context for use as trace data for the simulation.

5.3.3.1 Conceptual model of the real world case

Robinson (2007) has defined the conceptual model as '*a simplified representation of the real system*'. According to Law (1991), conceptual modelling is the most important element of the simulation process so the model plays a vital role reflecting the real system in a simple model in order to simulate it. Initially, a modeller spends time with users, understanding the actual system, then tries to build a simple model considering the scope of the simulation project.

In this research, the conceptual model reflects the customisation process in a real world case, selected as the baseline model in the evaluation process of the CCRD. The contextual review study shown in (section 5.3) was conducted to learn about and understand the real work of the selected case and to determine the key components of the real system such as its inputs, outputs, assumptions and process and the whole system, in order to use it in the building process of the simulation model.

5.3.3.2 Representation of the conceptual model

According to Robinson (2007), there are four ways to represent a conceptual model:

- Component list
- Process flow diagram
- Logic flow diagram
- Activity cycle diagram.

However, the selection of a representation depends on different factors such as goal of the model, the scope of the project and the available information.

In this research, the conceptual model has been based on the findings of the contextual review study (section 5.3). Therefore, the activity cycle diagram has been selected to represent the baseline model. Also, it was noted that the company in the contextual review uses a workflow system called JIRA[7] as project management software help to manage clients' requirements and other processes in the customisation cycle. This software provides a workflow diagram reflecting the current processes.

5.3.4 The designed conceptual model

A simple conceptual model has a number of advantages; according to Chwif, Barretto and Paul (2000), it can be developed more quickly, is flexible and is easier to interpret. Also, Chwif states that '*In general the aim should be to: keep the model as simple as possible to meet the objectives of the simulation study*'. The main goal of building a conceptual model of a real world case in this study is to simplify the current model, based on the scope of the research and its goals. Therefore, some processes in the current model have incorporated one or two components to make them readable and easier at the simulation stage. For example, the development stage in the real work has eight activities in two groups as the main goal of the research is to discuss the impact of changing the location of decision making on the customisation process, rather than discussing sub-activities such as testing, scheduling or verification.

[7] JIRA is a tracking system for multi teams projects (https://jira.atlassian.com)

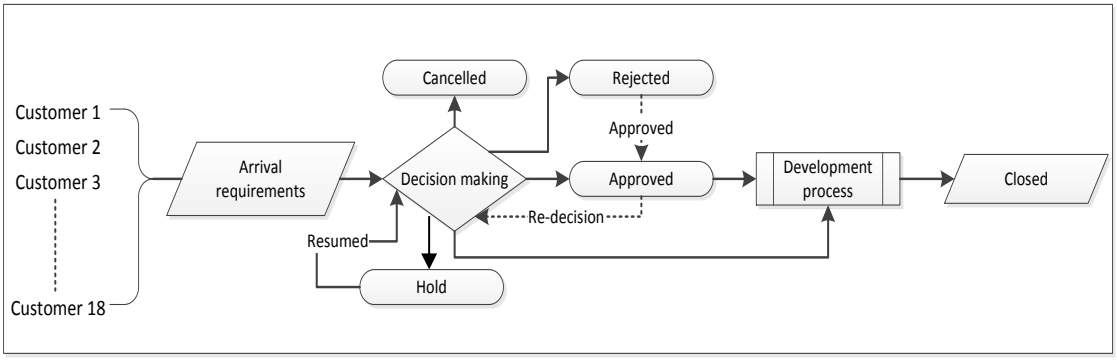


Figure 5.3: The conceptual model of the real world case

In the conceptual model of the baseline (Figure 5.3) the customisation process starts at the clients' locations. The clients issue their requirement themselves or through the implementer at the company. It comes into the queue awaiting a decision, then decision making takes one of fifth routes:

1. **Cancelled**: Cancellation happens if the client cancels the requirement, or the decision maker decides to cancel the requirement after discussing it with the client. In this process the requirement will be eliminated and closed immediately.
2. **Rejected**: Rejection happens if the requirement does not make sense. For example, it may have been developed before, out of the contract's scope or will have an effect on other systems. Any rejection needs time to discuss with the client to inform him or her of the reasons behind the decision. Although most rejected requirements are closed immediately, some are approved after discussion with clients and go direct to the approved requirements queue.
3. **Hold**: Some requirements are unclear and need discussion with the client, implementer or more investigation. The decision maker puts this type of requirement in the holding process, meaning that a decision has not been made and it will take time before it is returned to the queue for decision-making.
4. **Approved**: Once an approval decision is taken, the requirement goes into a queue for the assigned developer to accept the task, give an estimated time of completion and schedule it into his or her job list.
5. **To development process direct**: Some requirements such as for bugs or urgent cases are sent direct to the development process as they need neither confirmation nor estimates of time. These types of requirements go directly to the assigned developer's job list. Development process in the real world case, the development process has between six and eight activities including developing, testing and verification. In this conceptual model, all the development processes are gathered into a single activity termed 'development' that

starts with commencing the development process and finishes when the requirement is confirmed and ready to deliver. This process is unchanged in this study and to makes for a conceptual model that is as simple as possible.

5.3.5 Historical data

Robinson (2007) defines trace data as streams of data that include important information about the events and activities such as arrival time, time accrued and the nature of the requests. Also, he states that trace data are read by a simulation as it runs the event reflecting the real system. Using trace data to run a simulation model enables it to represent historic events of the real system, helping to evaluate the model and its performance. Therefore, the second part of the contextual interview study output comprises historical data on the clients' requirements. The company exported a historical log file of its customisation of clients' requirements covering the period from 23/03/2013 to 21/10/2013. These data did not include any confidential information, thus conform to ethics considerations, and were collected to use as trace data to run the simulation models in the evaluation process.

5.3.5.1 Description of the collected data

The selected company used JIRA as a project management system (see section 5.3.3.2) for tracking and managing clients' requirements. Significant efforts were made to clean the 'noise' from the collected data to prepare them as trace data to run simulation models. An example of this 'noise' was the due to the JIRA system being only from 23/03/2013, so earlier data had to be ignored as it did not have full information about clients' requirements such as arrival time, decision time, development time and resolving time; this lack of process detail meant that data could not reflect the real behaviour of the real system. However, they covered a random chunk of data, including different activities of clients and developers. The clean data focused on the completed requirements that had gone through the whole customisation process and it helped in calculating the entire process time from requirement creation to resolution. These data were used as trace data for the rest of the simulation experiments including the validation process of the baseline model, comprising 2479 requests for 18 distributed clients' requirements and covering the activities of 1290 working hours for nine hours each day, starting from 8:00am, excluding weekends and holidays.

In summary, the contextual review presents the main goals of this study, which were collecting information about the real case study from the selected company's workplace in order to design a conceptual model for its actual current customisation and to collect historical data for use in

designing the simulation, driving them as trace data. Furthermore, this section describes the case study and interviews conducted, with the major outputs of this study.

5.4 Simulation Method

This section presents the importance of modelling and describes simulation process in research and evaluation studies and how it reduces the effects of conducting experiments on real cases. According to Robinson (2007), simulation is '*Experimentation with a simplified imitation (on a computer) of an operations system as it progresses through time, for the purpose of better understanding and/or improving that system*'. Modelling and simulation have become popular methods in research and industry in many subjects such as engineering (Oden et al., 2006), business (Jahangirian et al., 2010) and medical science (Gaba, 2004), however they is used increasingly to support software development and software project management (Psaroudakis & Eberhardt, 2011). In addition, Kellner et al. (1999) state that software simulation has been used for various purposes such as the strategic development of software, process improvement, and control and operational management of software engineering.

5.4.1 Simulation for evaluation research hypothesis

In this research, simulation has been selected as the evaluation method for the research hypothesis. According to Abdel-Hamid (1988), although it is easy to propose a hypothesis in software engineering it is difficult to test it. Many reasons are behind this difficulty and applying and controlling software engineering experiments in real word cases involves costs and time (Setamanit, Wakeland & Raffo, 2007). However, the challenges multiply in a large, complex and dynamic project such as distributed development (Glass, 1982). Therefore, simulation is used to enable researchers to control software engineering experiments and to identify the factors that impact on the outputs of the simulation model at less cost and time (Setamanit et al., 2007). In terms of evaluation software development, Martin and Ra (2001) state that changes can be evaluated by simulation. However, evaluation should consider the project environment.

5.4.2 Discrete-event simulation approach

There are two types of computer-based simulation: continuous simulation, which refers to a computer model of physical systems and the tracking of the dynamic changes of the systems behaviour over time; and discrete-event simulation. The latter is one of the simulation techniques and strategies for designing a simulation model, relying on the occurrence of events

from state to state over time (Psaroudakis & Eberhardt, 2011). However, the characteristics of the system to be simulated help to choose the most suitable technique (Lunesu, 2013). According to Sokolowski and Banks (2009), discrete-event simulation is formally defined as the variation caused by a sequence of events acting on the model. Therefore, this type of simulation targets systems that involve state changes based on the occurrence of system, for example traffic lights changing from red to green to yellow. On the other hand, systems that have arrivals and services are known as a queueing system, representative of many features of our daily lives such as call centres and software development processes (Sokolowski & Banks, 2009).

Regarding the challenges of applying a new solution to testing software engineering research hypotheses in the real world, for instance the cost, time and availability of resources (Setamanit et al., 2006), simulation has been selected to model a real world customisation and then to apply the new ideas from that simulation to the rest of the research hypothesis. In addition, this research used discrete-event simulation to simulate the customisation process as it is appropriate for software development activities. The customisation process model has activities and areas for changing the state of the system, starting from the arrival point and then making a decision that takes one of several forms, and finally to leave the system at a specific time.

5.4.3 Software simulation package

According to Sokolowski and Banks (2009), computer software for simulations has featured continuous improvement, starting from the first computer simulation in the 1950s using a general programming language, then specific programming languages and software developed for simulation purposes. The past two decades have seen a significant jump in the number of viable simulation languages and simulators, since the PC (personal computer) and other technologies such as Windows were launched in this period (Sokolowski & Banks, 2009). Today there is a variety of available emulators to support and provide visualisations and interactive tools. A visual display of the simulated model has many benefits. Robinson (2007) lists some, as follows:

- Greater understanding of the model.
- Easier model verification.
- Enables interactive experimentation.
- Improves understanding of the results.
- Improves communication of the model and its findings to all parties.
- Provides the potential for using simulation in group problem solving.

In this research, a simulation package has been selected and used to build and run simulation models. Choosing a simulation package was a hard task. However, Robinson (2007) has summarised the steps and factors of selecting simulation software packages:

1- Modelling requirements.
2- Evaluation criteria.
3- Availability of the software.

The package selected for this research to build the simulation for the evaluation process is called SIMUL8.[8] It was developed by SIMUL8 Corporation and provides an integrated environment including a simulation modeller and a results analyser. Also, it provides a powerful language and model visualisation to enable users to create flexible and robust simulations. SIMUL8 has been selected in this research as it has been used widely in industrial and academic fields (Concannon et al., 2003). Using this package made the building of simulation models easier, as the models focused on an activities diagram supported by SIMUL8. The University of Southampton has a license for the student version, so using it made it easy to access a licensed simulation package.

5.4.4 Evaluation measurement and metrics used in this research

Fenton and Pfleeger (1997) state that 'measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules'. They have become commonplace in different engineering and scientific fields. Software engineering is an applied discipline of computer science, which describes various techniques for constructing software products. The process requires management and engineering practices such as planning, costing, analysing, designing and testing (Fenton & Pfleeger, 1997). On the other hand, engineering disciplines use methods and techniques that contain models and theories, conducting empirical validation processes for the outputs of these theories and models in terms of testing and evaluation (Munson, 2003). Software engineering is one of the disciplines that produces a product in the form of systems and software through a sequence of processes and activities. Measurements in software engineering occur at three levels: software products; the software development process; and software quality (Ebert et al., 2005).

I. Measuring software products:

A software product is a complete set of computer programs, procedures and other documents, and data developed to deliver specific functions (Solingen & Berghout, 1999). These products

[8] http://www.simul8.com/

can be measured through software products metrics that aim to measure their various parts such as requirement specifications, high and low levels of design, source code, testing cases and documentation in order to deliver valid software products (Munson, 2003).

II. Measuring software quality:

Software quality is defined as the characteristic of a product and its ability to fulfil the client's requirements to gain their satisfaction (Solingen & Berghout, 1999). Measuring quality in software products, and processes as well, is an important aspect of software engineering; however, it is a relationship between quality and other variables such as cost and time, as ISO standard 9126[9] refers to the areas of software improvement in quality: functionality; reliability; usability; maintainability; efficiency; and portability (Figure 5.4).

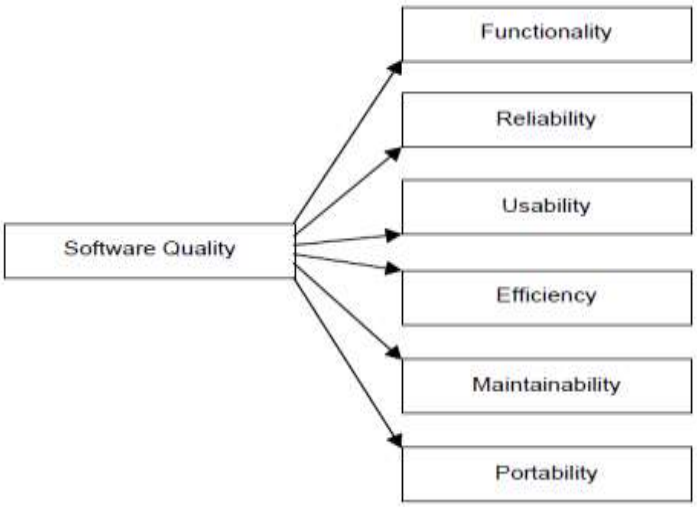


Figure 5.4: Software quality characteristics (ISO 9126, 1992)

[9] ISO/IES 9126, Information Technology - Software Product Evaluation - Quality Characteristics and guidelines for their use. International Organisation for Standardisation, 1991.

III. Measuring software process:

Software is produced by a sequence of steps and activities in the context of the software development lifecycle (Sommerville, 2011). A software development process is defined as the necessary activities to translate client requirements into a software product (Sommerville, 2011). Fenton and Pfleeger (1997) state that some software processes consist of the software development lifecycle phases of gathering requirements, analysis, designing, implementation and testing, though to the maintenance phase. Measuring software activities gives great insight into the adequacy of the time allowed to produce a software product and to evaluate alternative models and process (Munson, 2003). However, to improve the software development process the goals should be well-defined (Solingen & Berghout, 1999). Clear goals help to define and select the appropriate measurement and metrics to evaluate that improvement. The main goal of this research was to improve the customisation process to overcome the challenges of communicating customers' requirements across distributed boundaries. The implications of those challenges cause delay in resolving customisation requirements. On the other hand, the proposed model (the CCRD model) involves change to the locations of some activities. As in Scenario I it allocates decision making to the customer's location, and in Scenario II it allocates local development activity to the customer's location. These changes may impact on the duration of the decision making and development process and must influence the total time taken by the entire customisation. Therefore, measurements used in the evaluation process of this research are defined in terms of the goals of the research and the importance of measuring the productivity of development activities.

The design of the metrics used in evaluating this research aimed to measure the impact of the communication challenges on customisation in the distributed domain. There are many implications of distributed communication in software development projects, however the software lifecycle requires a great deal of communication and any issues directly affect productivity (Jiménez et al., 2009). One of the measurable factors of software development productivity is the delay time through the process. Herbsleb and Mockus (2003) refer to the delay in software projects and loss of time in development as the result of inadequate communication in the distributed domain. Further research conducted by Herbsleb and Moitra (2001) emphasises that the most frequent consequences of cross-site problems is delay in resolving software issues, that is, the additional time to complete a requirement when more than one site is involved. In addition, Azam et al. (2014) indicate the main areas of software process improvement, which are to decrease project cost, to decrease project risk, to shorten the project cycle time and to increase product quality (Figure 5.5). Indeed, Fenton and Pfleeger (1997) indicate that time is one of the measurable elements of software activity productivity.

In summaries, measuring delay time in these stages aims to assess it through the customisation process as an impact of communication challenges in the distributed domain, therefore the evaluation metrics for this research contains measurable elements as follows.

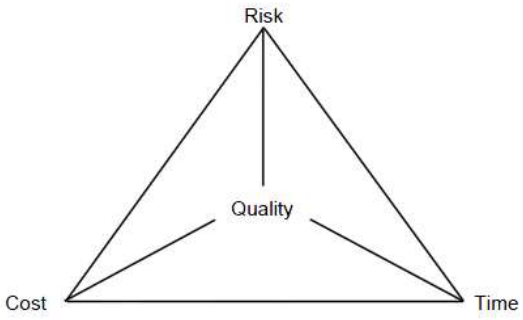


Figure 5.5: Areas of software process improvement (Solingen & Berghout, 1999)

- **Decision-making time**

Decision-making time is calculated from the time a requirement is created by the client to the time a decision is made. It includes the waiting time before the decision-making process and the transformation time from client to decision maker. The duration of decision making is selected as an evaluation measure in this research due to the impact of changing the decision- making activity location in the CCRD model from the central team to the distributed domain at a customer's location. Removing some activities such as holding and rejection may cause a difference in the decision time. In addition, challenges of communications in the distributed domain directly affect the duration of decision making and cause delays to production.

- **Development process time**

The duration of the development process is calculated from the time a requirement is approved by a decision maker until it is resolved, including the waiting time before the development process, development, testing and signing off as resolved. Using development time as one of the metrics elements in this research comes from the important impact of the development process on the customisation process in general, and on the CCRD model in particular. In the CCRD model with local development processes at the customer's location, a parallel development process applies, affecting the duration of the process and consequently the total time for the entire customisation process. As Rini van Solingen and Berghout (1999) indicate, using parallel development is one form of shortening the duration of the project cycle, which is an important key to process improvement.

- **Total time for the entire customisation process**

This is the duration of the whole process that requirements undergo for customisation purposes. This time is calculated from the creation of the requirement by the client to the delivery of the solution and acceptance by the client, including decision time, approval time and development time. It also includes the time spent in communication during the holding or rejection processes. The entire duration of the development lifecycle is an obvious numerical output of development model, which is an important aspect of measuring the productivity of models or methods in order to perform evaluations and assess the satisfaction of customers against the speed and time of response. This research uses the total duration of the entire customisation process, besides decision-making and development time, as its evaluation metric. It uses a validation of the baseline model (section 05.4.5.2) and the evaluation experiments in Chapter 6.

5.4.5 Baseline Simulation Model

In section 5.3, the contextual review study was conducted in a real case study in order to use it as a baseline during the evaluation process of this research hypothesis. One of the outcomes of the contextual review study was the conceptual model of the real case study (Figure 5.3). This section describes the simulation model of the real case study by transforming the conceptual model into a simulation model. Also, the verification and validation process is applied to the simulation model (as shown in section 05.4.5.2).

5.4.5.1 Constructing the baseline simulation model

During the contextual review of the real case study, much information has been collected in terms of the activities in the customisation process in that case. Although the collected information is used in designing the conceptual model, it is a non-software specific description of the simulation model (Robinson, 2007). Therefore, the model enhances the real case assumptions to reflect the real customisation process in the case study. The baseline simulation model shown in Figure 5.6 has several components, starting at the arrival point and ending with the development process. The fit of the model activities has been based on information collected in the contextual review, as mentioned at the beginning of this section, historical data exported from the company tracking system and all the information from the confirmatory study (Section 5.5). Furthermore, that setting of the simulation model has been checked and confirmed during the confirmatory study (section 5.5). The following sections describe the role and purpose of each component and the verification and validation process for the baseline model.

- ***Baseline model components***

i. ***Arrival point:*** The arrival point in this simulation model represents the arrival of requirements from different clients at the central customisation system. In this research, trace data (actual arrival time in the real system) have been selected as the arrival time for the simulation model in order to unify the inputs of the real system and the simulation model to compare them. Using actual data to drive the simulation model is recommended if the data are accurate, and may provide a valuable insight into and readability of the simulation (Law, 2007).

ii. ***Activity:*** In this model, there are five activities: decision making; rejection; holding; approval; and development. A description of these activities' roles and purposes is given in the section on the conceptual model (section 5.3.3.1) as outputs of the decision-making process. The setting of each activity includes probability distributions of service time, input and output routing from the historical data collected in the contextual study.

Regarding their distribution, in this research the distribution of the real system has been applied, obtained from the collected data. Fitting simulation model distribution is one of the hardest tasks in simulation building (Robinson, 2007), as any failure to choose the correct distribution can affect the accuracy of model output (Oden et al., 2006). It can be done in two ways, either by selecting one of the common probability distributions such as Poisson, Exponential or Uniform, or, in other situations where there is a need to use observed data to specify the appropriate distribution that fit a simulation activity, what is called in this case an *empirical distribution* (Robinson, 2007).

This study selected an empirical distribution to fit the distribution for the baseline simulation model to reflect the reality of the selected case study. In fact, the empirical distribution held some issues such as the challenge in applying the goodness-of-fit tests, as data have some assumptions regarding the effects of the accuracy of fitting (Oden et al., 2006). Also, in some situations it is not possible to collect data on random variables, or the collected data lack important data. Law (2007) states that '*sometimes data are collected by an automated data collection system, which doesn't provide the data in a suitable format.*' And that is indeed what happened in this research. The historical data of the real system had been exported from a tracking system that provided time of arrival and departure for each task, with a clear lack of service time and queueing time. This led to difficulties in specifying the service time for each activity in the real system. So, in this case, the fit applied was based on the general shape of the histogram of activity time including service and queueing times, as that described for the validation of the simulation model (section 0 5.4.5.2). This assumption in the simulation setting including distribution fitting has been discussed with experts from the selected case study, as

recommended by Law (2007), as a vital factor to complete the lack of data by selecting the suitable distribution.

In the five activities in the baseline model, the Beta distribution has been selected as it is recommended in view of the absence of data (Fente et al., 1999). Table 5.2 shows an overview of the simulation activities and their distribution.

Table 5.2: Overview of the simulation activities and their distributions

Activity	**Distribution name**	**Alpha 1**	**Alpha 2**	**Min**	**Max**
Decision making	Beta	1.3	14.7	0.01	4
Rejection process	Beta	0.1	1	0.02	60
Approval process	Beta	0.2	10.3	0.02	34
Holding process	Beta	0.1	1	0.02	40
Development process	Beta	0.3	24.5	0.02	26

iii. ***Queue:*** This component was established before each activity for queueing of the service.

iv. ***End:*** This is the end of the simulation model representing the end of the customisation process in the real case (resolving the issue and completing the requirement).

v. ***Resources:*** In the simulation model each activity has resources, which in this case are human resources. For example, in decision making there are in the real case study nine decision makers. Therefore, in the simulation model there is a resource component called 'decision resource' (Figure 5.6). It is defined with nine resources to run the decision-making activity. All resources have been based on historical data in the real system and then in the confirmatory study. Table 5.3 shows an overview of the activities and defined resources in the simulation model.

Table 5.3: Overview of activities resources in the simulation model

Activity	**Resource name**	**Number of resources**
Decision making	Decision	9
Rejection process	Rejection	5
Holding process	Holding	4
Approval process	Approval	17
Development process	Development	

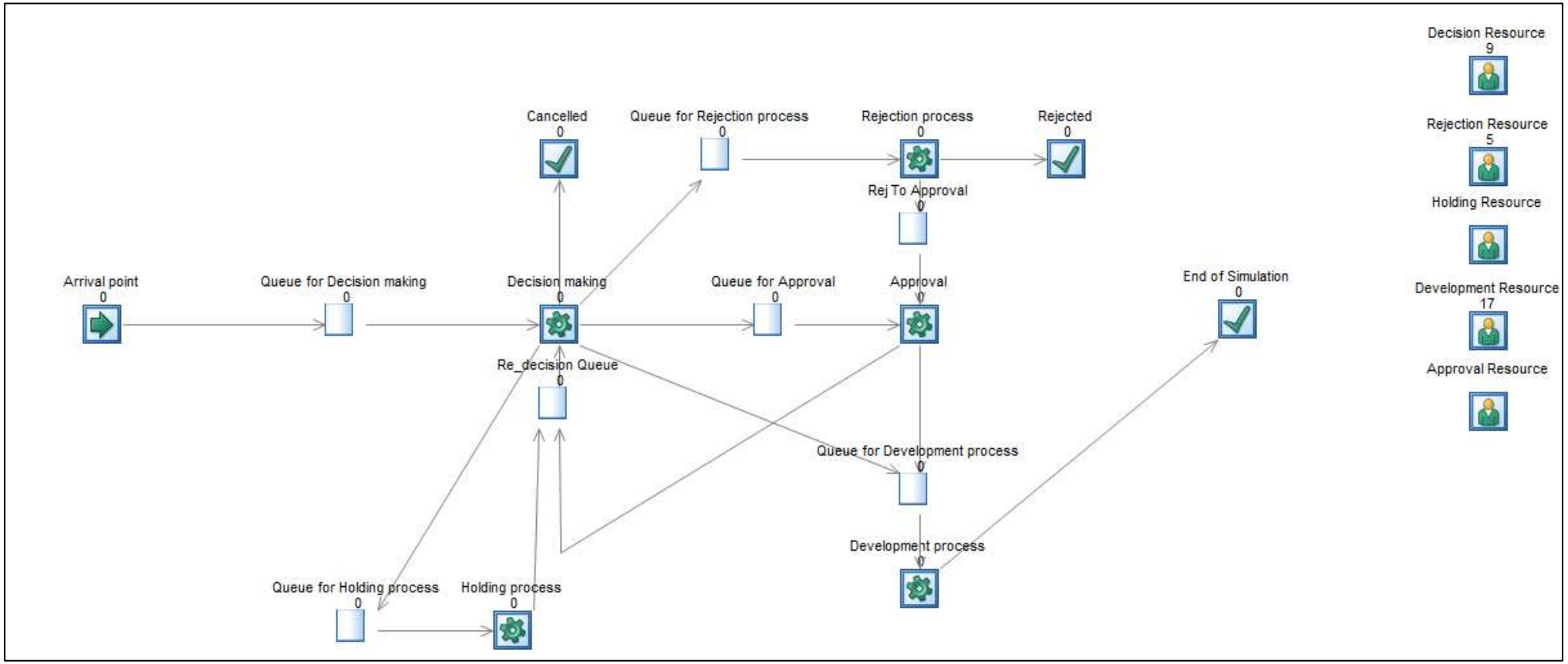


Figure 5.6: Baseline simulation model for the real case study

5.4.5.2 Verification and validation of the baseline model

Verification and validation are significant aspects in simulation projects, lending credibility and reliability to the simulation model and its outputs (Robinson, 1997). Davis (1992) defines the verification process of a simulation model as an process that ensures that the transformation of the conceptual model to a computerised simulation model is sufficiently accurate, while validation ensures that the simulation produces sufficient results to meet the purpose of that model (Kleijnen, 1995). This section presents the verification and validation process applied to the simulation model in this study, as follows.

- ***Verification process***

One of the most difficult issues facing a simulation project is how to present the real system in an accurate way (Robinson, 1997). According to Kleijnen (1995), there many ways to apply the verification process to the simulation model. In this research, two approaches have been applied in order to ensure the simulation model reflects the real system, the experts' review and tests the goodness-of-fit.

i. Expert's review

An experts' review is an approach suggested in some cases to build the simulation model and fit the distribution of its activities (Law, 2007). In this research, an experts' review was applied in two stages of designing the conceptual model and the structured of the simulation model through the contextual review study (section 5.3) and in the confirmatory study (section 5.5). The final product and the setting used in the model have been discussed and confirmed by seven experts at the real case study, at that stage.

ii. Testing the goodness-of-fit

A goodness-of–fit test is a statistical test used to assess formally whether the observed values of the selected distribution fit the values of expected distribution (Banks et al., 2004). There are two methods to check the fit. First is the graphical test, which compares the histogram shape of the selected distribution and the expected one and tries to select the suitable distribution. The second method uses a statistical test, there are tests for this purpose such as the Chi-square test and Kolmogorov-Smirnov test (Field, 2009). In some cases, a statistical test is not applicable and the graphical test and experts' review is recommended, especially for empirical distribution (Law, 2007). In this research, the empirical distribution has been selected, based on the historical data. Therefore, the statistical test does not work with the expected distribution as the actual system does not follow a model systematically in its activities. For example, the decision-making process does not follow any service model such as 'first in, first out' or by priorities. So, that affects the distribution of the service output. In addition, the historical data lack information

such as service time. Law (2007) refers to this issue as one of an empirical distribution problem. In this case, the fit of the distribution is applied using graphical tests on the graph of queueing and service time of the activities in the model, and fits that time on the actual system graphs as shown in Figure 5.7. Also, some information was obtained from experts such as minimum and maximum times for services.

Fitting distribution (especially for empirical data) is not a single step, and should comprise a number of iterations through different stages to select and confirm the appropriate distribution and assumptions used (Robinson, 2007). Therefore, the setting used in the simulation of the baseline model and of the CCRD model has gone through iterations between the simulation building stage and the confirmatory study to check the assumptions used with the experts. For example, in questions on the confirmatory study (section 5.5) there were some results from the initial running of CCRD simulation models. These results have been reviewed by interviewees at that stage who provided various reasons for the wrong assumption, which was that the historical data had a number of requirements termed ATSE that were created internally and had very low priority. Those requirements affected the queueing time of customisation activities and gave different distributions. As a result of that review, the assumptions and distribution of activities have been repeated and confirmed. So, the distribution assumptions mentioned and used in previous sections, also the results and setting of the evaluation experiments in Chapter 6, are the accurate and final data.

- ***Validation process***

In order to ensure the simulation model provided accurate results, a validation process was undertaken. One of the most used validity tests for a simulation model is comparing the output of the simulation model with the output of the actual real system, called *result validation* (Banks et al., 2004). There are three types of comparison validation for the simulation model: comparison with an existing system, comparison with experts' opinion and comparison with alternative model (Law, 2007). This research used two of these to test the validity of the simulation model. The comparison test with the real system of the selected case study has been applied following the method shown in Figure 5.8 through the 'black box' technique (Figure 5.9), using the same historical arrival time for both the real system and simulation and comparing the outputs. Also, the experts' review method was used through the confirmatory study (Section 5.5) to confirm the findings and how they matched the reality.

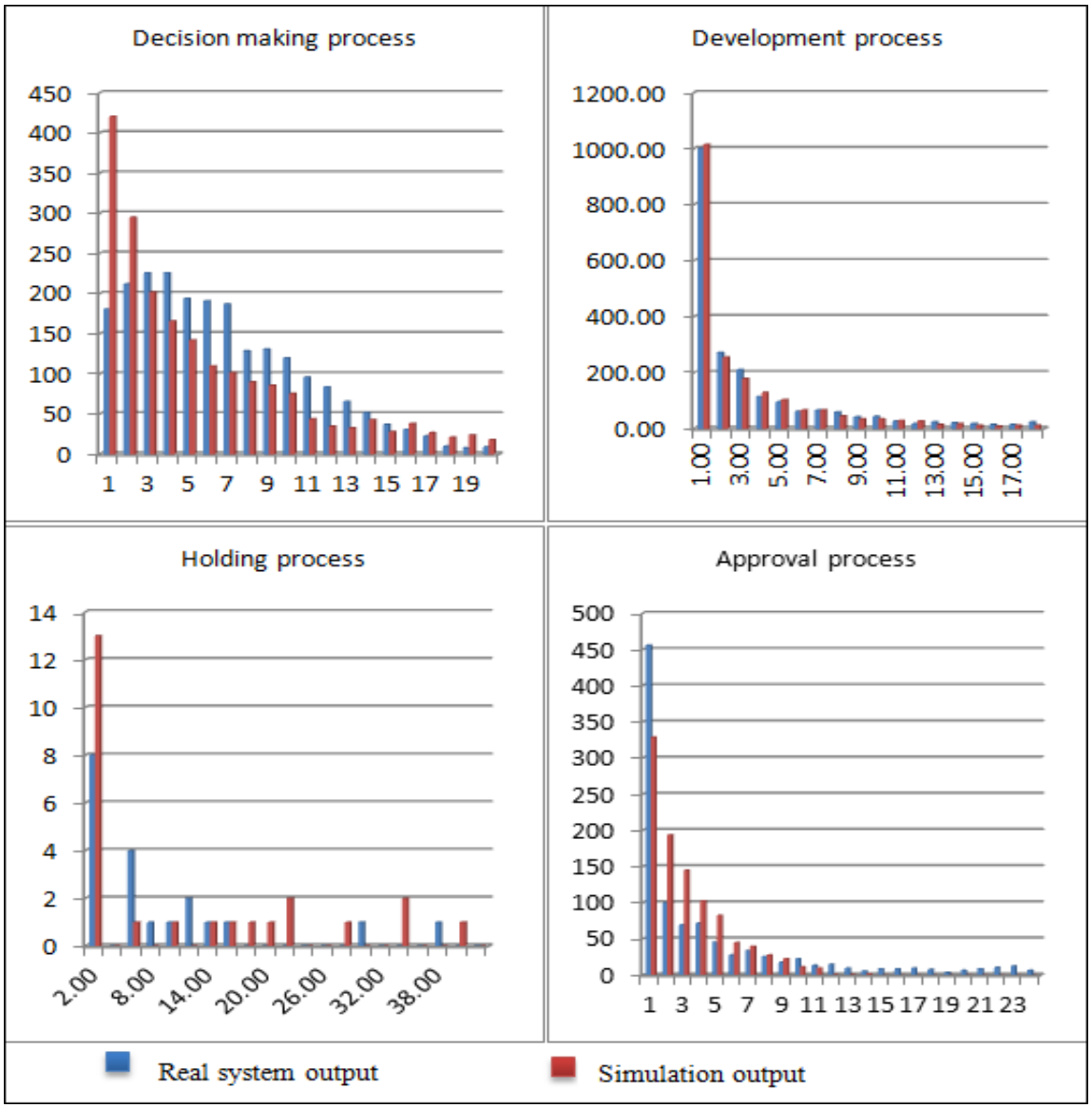


Figure 5.7: Graphical goodness-of-fit test for some activities

In order to apply the 'black box' validation test, a statistical test was used to find if there was a significant difference between the output of the real system and the output of the simulation model. Although the appropriate statistical test for this validation comparison of two groups is Student's t test (Argyrous, 2011b), this emphasises normality of the data as one of the required assumptions. In this research, the collected outputs of the simulation model and the real system are skewed, which is not normal. Figure 5.10 illustrates some examples of the real system and simulation outputs. In this case the available directions to complete this comparison are data transformation or using a non-parametric test (Field, 2009).

i. *Transformation of the data*: transforming obtained data to normal and applying the t test is not recommended in some cases (Field, 2009) as it describes data of different formats based on the type of transformation and in some cases it does not work, as in this study.
ii. *Using non-parametric test*:

With empirical data, often the data have a different distribution and it is not always easy to correct the problem in the data to transform to a smooth normal distribution (Field, 2009). In this case, the non-parametric test was used as it does not require many assumptions such as normality of the data. It relies on a ranking approach to test the significance between two groups.

In this study the purpose of applying this test was to compare the outputs of the real system and the outputs of the simulation in different activities to ensure the simulation model provided accurate results. The Wilcoxon signed-rank test has been applied, which is equivalent to a paired t test. The Wilcoxon signed-rank test is similar to a dependent t test; however, it uses a ranking approach to compare the two groups of data (Field, 2009: 540). The Wilcoxon signed-rank test was run on each activity in order to compare the actual output of the real system and the simulation model, considering the following hypothesis with a significant level (P value) = 0.05 at the 95 per cent level of confidence, as typically used in human experiments (Argyrous, 2011b).

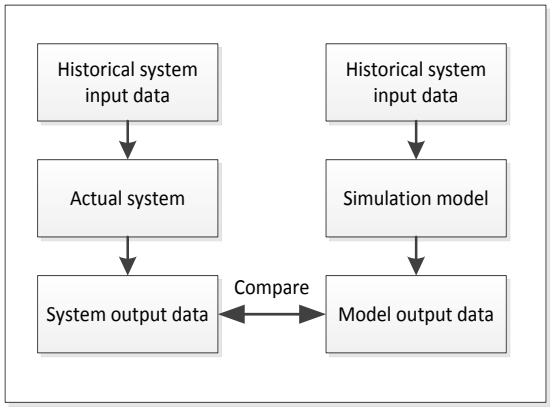


Figure 5.8: Correlated inspection approach (Law, 2007)

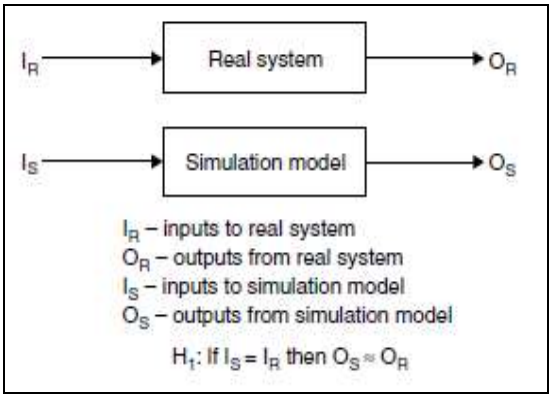


Figure 5.9: 'Black box' validation test (Law, 2007)

Hypothesis 0: *There is no significant difference between the real system output and the simulation output.*

Hypothesis 1: There is a significant difference between the real system output and the simulation output.

Therefore, when *P* <0.05 the *null hypothesis* will be rejected and the *alternative hypothesis* accepted, which means there is a significant difference between two groups. On the other hand,

if $P > 0.05$ the *null hypothesis* is accepted and that means there is no significant difference between the two groups.

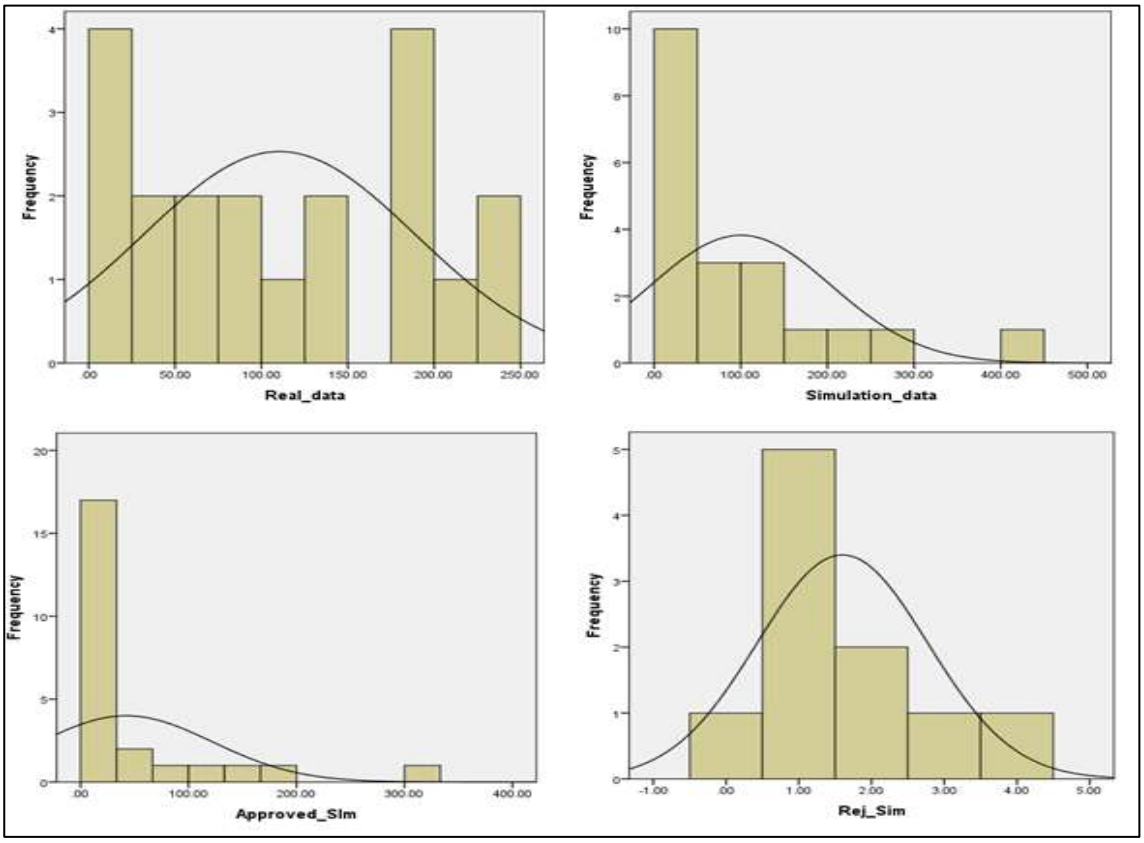


Figure 5.10: The normal curve on the histogram of the outputs of some results

- ***Results of validation tests***

Table 5.4 shows the results of the Wilcoxon test to compare real system outputs and simulation outputs. This table contains the results of the comparison for five activities. As is clear from the results, all P values for the five activities were greater than 0.05, which means that the defined null hypothesis is accepted, confirming that there was no significant difference between the outputs of the real system and the simulation model. Therefore, the results of validation test referring to the baseline simulation model provide results as expected and reflect the real system. Furthermore, the results obtained were presented to the participants in the confirmatory study to gain their opinion on it. They confirmed the results and gave positive feedback, as shown in the findings of the confirmatory study (section 5.5).

In summary, the simulation section presented simulation modelling techniques and tools used in this research. Also, it provided a description of the simulation of the selected case study in order to use it as a baseline model in the evaluation experiments of the CCRD model. Furthermore, the baseline model would provide a reasonable simulator for the customisation process as it was

based on actual data for a real industrial case study, especially useful in view of the scarcity of simulators of customisation for the distributed domain. While this section presents the simulation method and process used in this research including a statistical test applied in validation process of simulation model outputs, the setting of the simulation model of CCRD and results are presented in the chapter on findings (Chapter 6).

Table 5.4: Results of the Wilcoxon test for real system outputs vs simulation outputs

Activity	**Model**	**N[1]**	**Mean[2]**	**Std dev**	**P value**
Decision process	Real system	20	100.05	104.31	0.167
	Baseline	20	116.40	109.08	(P >0.05)
Holding process	Real system	25	0.92	1.73	0.953
	Baseline	25	1.00	2.58	(P >0.05)
Rejection process	Real system	10	1.20	2.09	0.395
	Baseline	10	1.60	1.17	(P >0.05)
Approval process	Real system	16	109.41	109.41	1.00
	Baseline	16	105.55	105.55	(P >0.05)
Development process	Real system	17	124.53	273.22	0.449
	Baseline	17	124.52	173.90	(P >0.05)

[1] Number of intervals [2] Mean of events in each interval *Sample size 2479 requirements

5.5 Confirmatory Study

In research methodology, there are two approaches for conducting a study and defining a hypothesis, the inductive and the deductive. The inductive process is defined as a bottom-up approach that moves from specific observations to generalisations toward specific theories (Gray, 2009). The deductive process works in the opposite way and moves from more general information to more specific conclusions, called the 'waterfall' approach or top-down approach (Gray, 2009). Researchers taking this approach start with a theory and narrow it down to test that hypothesis.

This research began with various hypotheses developed after reviewing the literature, as defined and presented in Chapter 1, and the evaluation process aimed to evaluate and confirm the defined hypothesis. This confirmatory study was designed to link the evaluation process through simulation with the opinion of experts in a real world case study. In addition, designing a simulation model for real systems through empirical data needs iterative confirmation process to ensure the simulation model of the real system reflects reality as much as possible (Law, 2007).

So, this study also aimed to confirm findings in the contextual review study and to present the simulation model of the real system in the selected company (section 5.3) to establish experts' opinion on that model.

The confirmatory study was designed to achieve the following objectives.

1- Check the finding of the contextual review study, which were the conceptual model of the current customisation model in the selected company, and the findings of the historical data collected from the company and used in the setting of the simulation of the baseline model.

2- Confirm challenges of communication customisation requirements across distributed boundaries, and the impacts of these challenges on the duration of decision making, development process and total time taken for the entire customisation process.

3- Gain the opinion of some experts in the company regarding allocating decision making and some development process to the clients' location, and how much that would help to reduce the implications of communicating customisation requirements across the distributed domain.

In software development, the evaluation process usually requires a detailed description of a particular system and its activities or the people who use this system (Patton, 1987). That level of information may come from observations, interviews and reviews of documents, as the qualitative method provides deep meanings, especially if the process includes human factors such as developers or clients (Creswell, 2008). This study was conducted using qualitative methods to achieve the objectives mentioned, through the semi-structured interview technique.

5.5.1 Using semi-structured interviews

Conducting research in a real workplace, collecting data or investigating an issue for a specific purpose requires particular techniques. An interview is a one of the most common and is used in various ways for particular purposes (Denzin & Lincoln, 2005), defended in a qualitative context as the interviews are conducted by a researcher in different formats, such as face-to face, by telephone or in focus groups, and may involve structured questions and general open-ended questions to gain the opinions of participants (Saunders et al., 2009). A qualitative research method is designed to look at hidden and unknown scenarios (Marshall & Rossman, 1999). Gray (2009) states that the interview is the most logical research technique for a number of situations. In addition, he claims that interviews are useful if conducted in a workplace to give people space to talk about their work and allow them the opportunity to reflect on their experience and give their opinion without having to commit themselves in a closed technique, such as a questionnaire or a written document. At the same time, Cohen and Manion (1997)

point out the various purposes such as gathering information about a person's knowledge, testing a hypothesis and, in conjunction with other research techniques, to achieve a particular goal. There are several different types of interview such as structured interviews, semi-structured interviews, non-directive interviews, focused interviews and informal conversational interviews and the choice depends on the purpose and the objectives (Gray, 2009).

Hove and Anda (2005) state that many phenomena in software development are qualitative in nature and the appropriate technique to collect data about these phenomena is semi-structured interviews. Also, they refer to how interviewing people provides insights into their world, their opinions and their feelings. In addition, software development is involved with human factors in its practices during development, and the qualitative approaches through interviewing aid eliciting different opinions from various practitioners in the software development lifecycle. Semi-structured interviews are defined as those conducted by a researcher who could apply the interviews in varying ways from one interview to another, such as changing questions, omitting some or giving more explanations, including other interviewees' opinions (Saunders et al., 2009). Therefore, in this research semi-structured interviews have been chosen as a method for collecting data from the real case in order to achieve the confirmatory study goals. In addition, the selected case study has many processes with different people working on them such as decision making, development and system analysis, which need an adjustment to the question based on the interviewees' position, giving them specific explanations to obtain their opinion.

5.5.2 Design of the semi-structured interviews

With regard to conducting the confirmatory study in this research through the semi-structured interview approach, the design went through several stages. The following sections discuss the aspects adopted for participant selection in terms of conducting interviews, as Miles and Huberman (1994) identified:

- **Setting:** Indicates the place where research will take place.
- **Actors:** people will be involved in the interviews or observed.
- **Events:** what the actors will do in this research, for example conducting the interview.
- **Process**: The evolving nature of the conducted study by the actors within the setting.

5.5.2.1 Setting

The contextual review (section 5.3) was conducted at a software development company in Saudi Arabia and the outputs of that stage were historical data for their process, with 18 clients and 1290 observed hours. A conceptual model of the real world case was used to design and drive

the baseline simulation model. In this stage, the same company (explained in section 5.3.1) was selected for the confirmatory study in order to use the same parameters and settings as the contextual review. Also, it was easy for participants to understand cases and data from their workplace, in order to provide useful and valuable information on the challenges and barriers identified in the previous contextual review and the results from the simulation experiments to reduce the impact of communicating clients' requirements challenges across distribution boundaries.

5.5.2.2 Actors

Choosing the participants in qualitative research is purposeful rather than random (Miles & Huberman, 1994), due to qualitative study being designed to acquire a better and deep understanding of a problem or to obtain the opinion of the most appropriate participants. Therefore, selection of the participants in this study focused on the people with good experience of different practices during customisation projects, such as dealing with clients' requirements, making decisions and development. The power of purposeful sampling in qualitative research lies in selecting information – rich cases, in depth – meaning participants can provide in-depth information based on the purpose of the study (Patton, 1987). In this study, the CEO chose participants who were working on the processes discussed in the interview's questions as well as for their experience of the customisation processes.

As this research discusses the customisation process across distribution boundaries, the participants were selected from the central team of customisation and the representatives of the company at the client's location in order to obtain the views of both people involved in the central team and those located at clients' locations to deal with their requirements and communicate with the central team. Table 5.5 shows the participants in this research and their positions in the company with some information regarding their experience.

5.5.2.3 Events

The purposes of these interviews are to as follows.

1- Check and confirm the findings of the contextual review study (section 5.3), which were the designed conceptual model of the selected case and analysis of the collected information including the current model's challenges and data used in the setting of the simulation experiments.
2- Discuss the results obtained from the initial results of the simulation experiments and participants' opinion regarding the concept of changing the locality of the distributed customisation process through the CCRD model in both its scenarios, of local decision

making and local development processes, and the impact of that change on the communication challenges in the distributed domain and on the productivity of customisation process. Therefore, the interviewees were asked individually at one-to-one meetings the questions shown in Appendix B. The semi-structured interview was designed to ask the interviewee the question and to explain it, if it needed clarification. As this type of interviews has open-ended questions based on the information provided by the interviewees (Hove & Anda, 2005), sometimes other questions were asked or the comments of previous interviewees mentioned during the interview in order to obtain valuable information regarding the topics.

Table 5.5: Participants' information in the confirmatory study

Interview	**Position of interviewee**	**Description**
Int. 1	A technical development team leader	He is the leader of one customisation team in the company. Also, he is a decision maker.
Int. 2	A technical development team leader	He is a leader of customisation web application team. Also, he is a decision maker.
Int. 3	A developer	Experienced developer. He has 9 years' experience in customisation different customisation projects.
Int. 4	A developer in web section	A web developer who deals with different distributed clients in the customisation process.
Int. 5	An implementer	He has good experience in customisation process and dealing with clients' requirements. He plays the role of communicator between different clients and the central customisation team.
Int.6	A system analyst	He has huge experience in clients' requirements as he used to work as a representative of the company at a different location. Now works with the central team to deal with clients' requirements in some situations to understand it and negotiate it with different people in the development lifecycle.
Int.7	A developer	He is a developer in the central team. He has more than 5 years' experience in software development.

* Int = Interviewee

5.5.2.4 Process

Before conducting this study, initial contact was made with the CEO at the selected company to arrange the timing and to explain the objectives and the process of the study. After that, approval was granted by the ERGO committee at the University of Southampton as its ethical strategy has a significant importance in an empirical study, including qualitative studies (Gray, 2009). The ethics reference is 10009 (Appendix A). This study was designed to be anonymous, so its questions did not require personal information from interviewees or the company. However, all the collected data were kept confidential and used for research purposes only.

The interview started by explaining the purpose and process. At the same time, a consent form was provided for interviewee to sign to indicate the steps that had been taken to obtain permission in order to protect their rights, as that is most important in qualitative research (Marshall & Rossman, 1999). The interviews were recorded by a voice recorder after participants agreed to its use, as suggested by Hove and Anda (2005), in order to focus on the session and to interact with interviewee; after the recordings were transferred they were deleted. Later, the questions were read to interviewees and explained, if there was the need, using examples to make the purpose as clear as possible or to show them some historical data and cases, as suggested by Creswell (2008). In addition, the answers were discussed if there was a need to obtain more information on each question.

5.5.3 Analysis and interpreting of the interview data

The analysis of qualitative data is the process of giving order to the data, classifying them into categories and themes, whereas interpretation involves more explanation, focusing on various meanings and looking for relationships and links among the participants' answers and defined themes (Patton, 1987). Therefore, the analysis and interpretation of the collected data from the conducted semi-structured interviews in this study takes a sequence of the following four processes shown in Figure 5.11.

***Step 1*: Organise and prepare the data for analysis**

This step started immediately after interviews were completed, involving making notes and extracting quotations from the interview data, then translating from Arabic to English, as the interviewees preferred to use documents, including questions and terms, in English, yet to speak in Arabic to be clearly understood.

***Step 2:* Design categories and themes for analysis process**

In this stage four themes were defined to categorise and analyse the interview data. Those themes were based on the main goals of the confirmatory study:

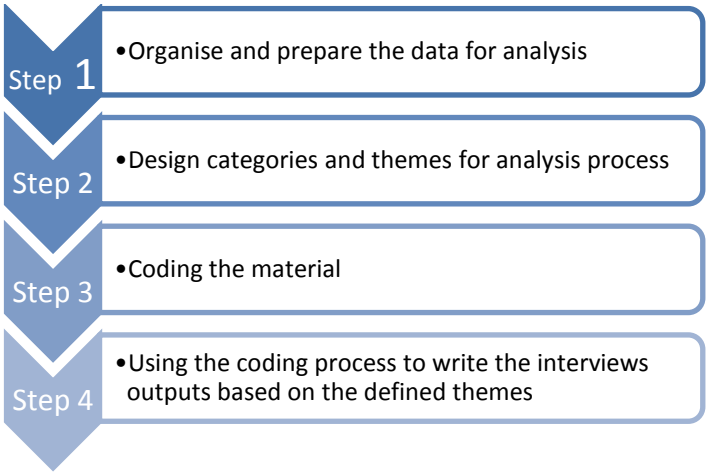


Figure 5.11: Process of analysis and interpretation of conducted interviews

1- Conceptual model of the current customisation model in the case study

This category aimed to collect data that confirmed or discussed the conceptual model of the current customisation process in the company, also the extent to which the simulation of the baseline model reflected the real system. This included the settings and assumptions used in the simulation models.

2- Challenges of current customisation model in the case study

This theme discussed the challenges of communicating the customisation process across the distributed domain in the current model and how much these challenges caused delays in decision-making time, development time and the total duration of the entire customisation process. In addition, this theme was designed to collect any data regarding the holding and rejection processes, which cause a clear delay in the customisation process.

3- Local decision making

In this theme interviewees' opinions regarding the CCRD model on making a decision at a client's location included their concerns and suggestions.

4- Local development process

This theme was defined to collect interviewees' opinions about the CCRD model regarding employing various development processes at the client's location, including their concerns and suggestions.

Step 3: **Coding the material**

Coding is the process of organising the material into parts or segments of text before bringing meaning to information such labelled categories (Rossman & Rallis, 2012). Codes, labels and categories should address something that readers expect, surprising information, unusual facts or a larger theoretical prospective in the research (Boyatzis, 1998). In this research, the categories defined in a previous step were used to code the data of the conducted interviews. NVivo was the software package used to complete this coding process, and is a qualitative data analysis tool designed by QSR (Jemmott, 2008). Afterwards, the related chunks of data in the interviews were linked to the appropriate category to make writing and analysing easier.

Step 4: **Using the coding process to write the interview outputs based on the defined themes**

The last step in this process was writing up the findings of the interviews and conveying that data with some interpretation to the reader. In the relevant section of Chapter 6 on the findings of the confirmatory study, the data of the interviews take the form of quotations and notes based on the defined categories (as explained in step 2) and explain the data with some interpretation of the important aspects of each part.

In summary, the confirmatory study conducted at the selected company (explained in section 5.3.1) was to confirm the findings of the contextual review study and the simulation of its current customisation model, the baseline model, used in the evaluation experiments in this research. In addition, it was to obtain the opinion of experts and practitioners in the company on the allocation of decision making and development process to the clients' locations. The confirmatory study was conducted through semi-structured interviews with seven participants working on different activities in customisation, including decision making and the development process. The findings of this study are analysed and explained in Chapter 6 using a thematic analysis technique based on the goals of the study.

5.6 Chapter Summary

This chapter presented the methods used in this research in order to build the environment for the evaluation process and to complete it for the CCRD model to examine the defined hypothesis for this research. This methodology started by selecting the case study used to provide a baseline model for this research, as well as historical data to run the simulation models built for evaluation purposes. The selected case study was discussed with the contextual

review study, conducted in order to collect information relating to customisation and its challenges in communicating customisation requirements in the distributed domain and to collect historical data to build and run evaluation simulation models (section 5.3).

Afterward, the customisation model of the selected case study was simulated for use as a baseline model for evaluation experiments for the CCRD model. Section 5.4 presented information about the simulation method and type used in this research, and the setting and assumption of the baseline model. Through the simulation, the baseline model for the evaluation process in this research was built and confirmed. Also, the simulation of the CCRD model for local decision making and local development processes was built and confirmed. However, the setting and results of those simulations of CCRD model are presented in Chapter 6, on the evaluation experiments and their findings.

Finally, this chapter presented a section on the confirmatory study conducted with seven experts at the same company as the previous study, to confirm the findings of the contextual review and the simulation process of baseline and CCRD models. Also, this was to obtain the opinion of experts from the case study regarding allocating the decision making and development processes to clients' locations, as the CCRD model proposed. The section presented the methods used in this study and the techniques applied with an explanation of the process, events and analysis approaches used; the findings of the confirmatory study are presented in Chapter 6, section 6.4.

Chapter 6: Evaluation Experiments

6.1 Introduction

In previous chapters the challenges and implications of communicating customisation requirements across distributed domains were discussed, and in Chapter 4 two scenarios for communicating customisation requirements in distributed domains (CCRD) model were presented to overcome the implications of these challenges. Chapter 5 outlined the methods selected for this research to construct and evaluate the proposed solution. In addition, Chapter 5 discussed the proposed case study and the simulation of its conceptual model, designed and built to evaluate the CCRD.

The main objective of this chapter is to discuss the evaluation of this research, including the methods used, the design of the evaluating experiments and the results obtained by each experiment. The process involves different methods to test the validity of the CCRD model and the extent to which it confirms the hypothesis, in order to answer the research questions. The following experiments aim to evaluate the CCRD model for local decision making (Figure 4.1) and for local development (Figure 4.2), by simulating the models and evaluating them against the baseline of the selected case study (section 5.3.5). Section 6.4 is a section presenting the findings of the confirmatory study. Figure 6.1 shows the structure of this chapter and the goals of each section.

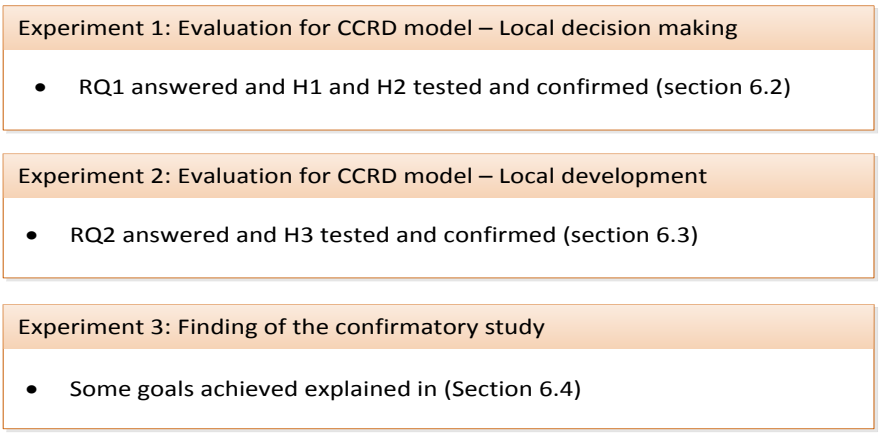


Figure 6.1: Structure of the CCRD model evaluation experiments

The questions the experiments aimed to answer were the following.

RQ2: What is the impact of making decisions locally on the total decision period and the total customisation period of software customisation processes in the distributed domain? (Answered by Experiment 1)

RQ3: How much would undertaking the process of development of minor tasks at clients' locations reduce the total development and customisation period? (Answered by Experiment 2)

In addition, the three hypotheses that the experiments tested were the following.

Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients. (Tested by Experiment 1)

Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients. (Tested by Experiment 1)

Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process. (Tested by Experiment 2).

6.2 Experiment 1: Evaluation for CCRD model – Local decision making

This experiment was designed to evaluate the CCRD model (Chapter 4) that employs local decision making on customisation requirements in the distributed domain. The experiment aims to test Hypothesis 1 and Hypothesis 2 (section 6.1), regarding how much the model reduces the impact of distributed-domain customisation communication issues on the time taken for decision making and the entire customisation.

6.2.1 Experiment design and methodology

In Experiment 1, simulation was used to achieve the objectives explained in section 5.3.1. In Chapter 4 the conceptual model of the CCRD (Figure 4.1) was built, reflecting the proposal to use local decision making to eliminate processes such as rejection and holding from the baseline model (Figure 5.3). This experiment began by constructing the simulation model for the CCRD

conceptual model using local decision making, as shown in Figure 6.2. The simulation of the CCRD model has the same settings as the baseline simulation model (Figure 5.6) in terms of its distribution of activities, percentages of decisions and resources. However, the activities' location and resources distribution have changed, based on the needs of the CCRD model.

In the simulation of the model shown in Figure 6.2, there is a representative from the software vendor at each client's location who takes the decision-making role. The customisation requirements process starts with the issue of a request by a client, which is discussed and negotiated face-to-face with a decision maker, with agreement on a decision taking one of three forms:

- **Cancellation:** when clients understand a problem with their requests and agree to cancel it.
- **Rejection:** when the request will affect other parts of the system, or when applying that request needs a further contract.
- **Approved:** when the request is accepted and is sent to the central team.

The central team receives requests from different clients and classifies them according to type; bugs requirements are sent directly into the development process, while new features requirements go through a further process to confirm the requests, then are either sent into the development process or returned to client's location for a new decision.

In terms of running the simulation model, the collected data from the selected case study in the contextual review (in section 5.2) featured the arrival requirements for 18 clients at an arrival point in the baseline model (section 5.3.5). In this experiment, the clients were divided into three groups based on their requirement numbers over 1290 hours (the observation time). Figure 6.3 illustrates the client groups and the sequence of conducting the experiments for each group. There are two reasons for this design. First, dealing with 18 clients in a single graphical simulation model poses challenges such as difficulty in visualising clients' locations and the central team location plus all the activities on a single manageable screen. The second reason is that applying the experiments to different groups is useful to see the effect of the proposed model on contrasting ranges of requirement numbers.

The same arrival data, that is, the clients' requirements, were input into the baseline model and the CCRD simulation model, and then the output from both models were compared using the metrics defined in section 5.4.4).

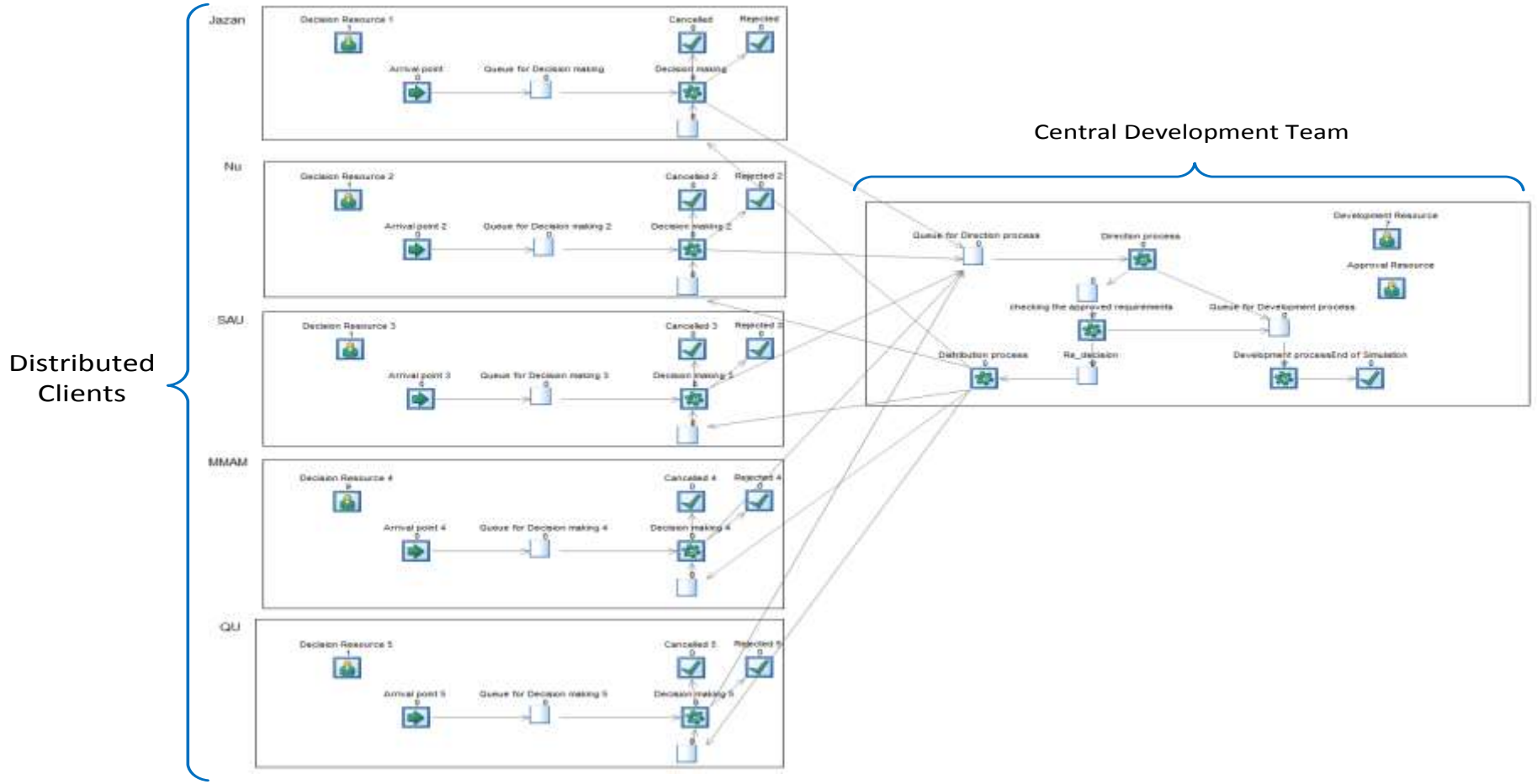


Figure 6.2: Simulation model for CCRD – local decision making model

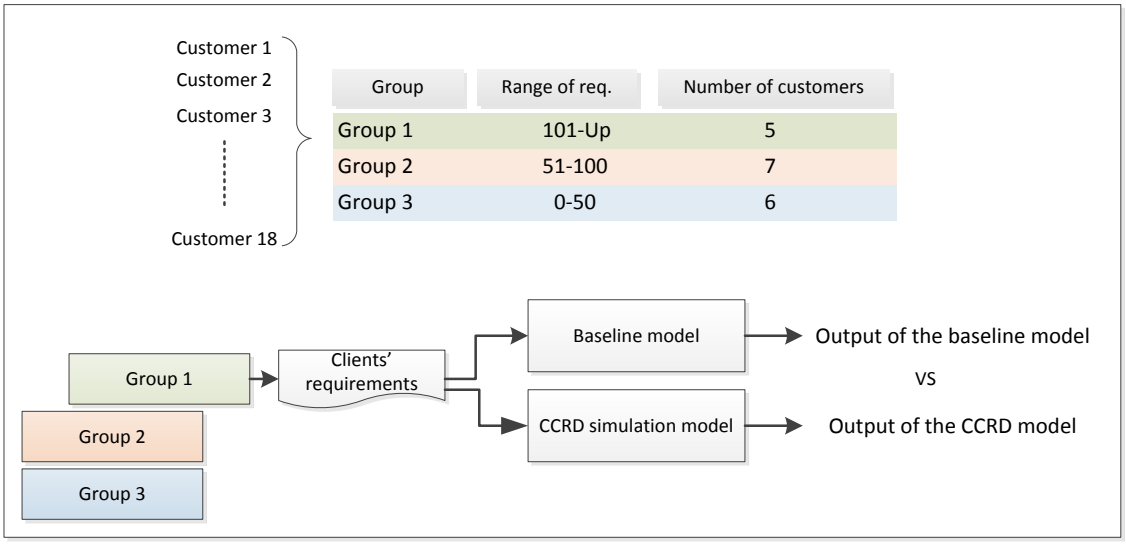


Figure 6.3: Sequence of conducting the experiment on different arrival data groups

6.2.2 Results and analysis

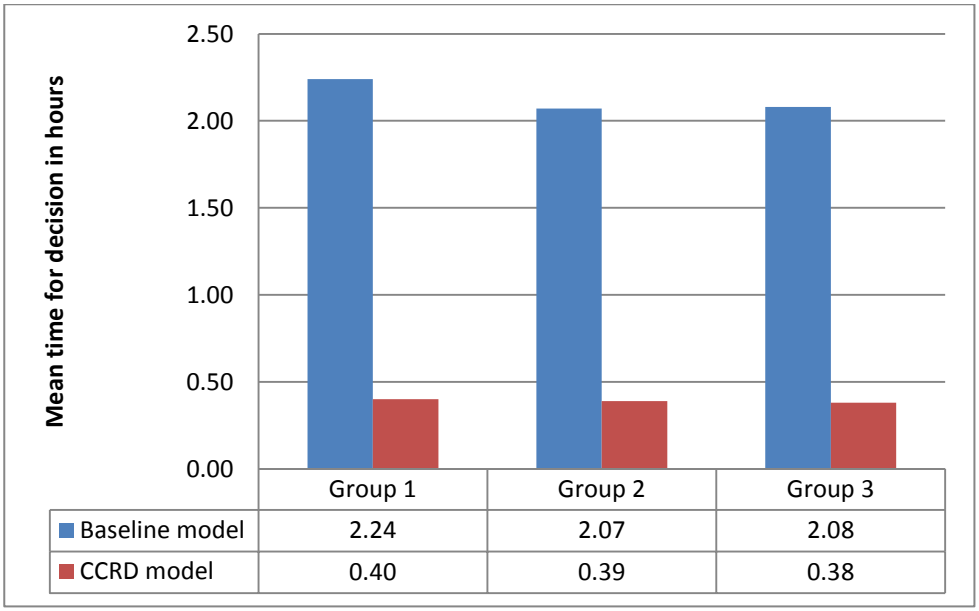
The metrics defined in Chapter 5 measure the decision time, development process time and entire customisation process time. Also, the evaluation experiments are designed to use these metrics to compare the baseline model and the CCRD model in order to evaluate the extent to which the CCRD model reduces the implications of communicating issues during customisation for multiple clients across distributed boundaries.

6.2.3 Decision-making time

In terms of results, the experiments aimed to measure the decision-making time, since one of the main activities was measuring the productivity of the customisation process model, as discussed in Chapter 5. Figure 6.4 presents the mean duration of decision making in the baseline model and CCRD model when local decision making is applied. In that figure, the simulation model runs three different groups with different client requirements, presented along the x-axis. The mean duration of decision making is an hour, presented on the y-axis; it takes all groups more than two hours under the baseline model, while the same requirements just take just under 30 minutes using the CCRD simulation model.

Table 6.1 shows the statistical data of the Wilcoxon non-parametric test, applied to test the significance of the difference between the output of the baseline simulation model and the CCRD simulation model.

Table 6.1 displays the mean duration of the decision-making process in the baseline model: 2.24 hours. The same requirement, that is, 1055 requirements for five different clients, takes only 0.4 hours under the CCRD model (about 24 minutes). Also, the mean duration of decision making for Groups 2 and 3 is 2.07 and 2.08 hours on the baseline, while under the CCRD model it takes 0.39 and 0.38 hours for the 877 requirements of seven clients in Group 2, and 337 requirements of six clients in Group 3. In the same table the null hypothesis, that there is no difference between baseline simulation model and CCRD model in decision-making duration, is rejected as the P value is <0.05; the alternative hypothesis is accepted, that there is a significant difference at the 95% level of confidence, for all three groups in this experiment.



	Group 1	Group 2	Group 3
Baseline model	2.24	2.07	2.08
CCRD model	0.40	0.39	0.38

Figure 6.4: Mean duration of decision making under baseline and CCRD models

6.2.4 Entire customisation process time

Changing any activity in the customisation process will affect the duration of the entire process. This section presents the results of comparisons between the baseline and CCRD models in order to measure the impact of applying the CCRD model to decision-making time and the entire customisation process, as this section discusses. Figure 6.5 illustrates the mean duration of the entire customisation process under the baseline model and the CCRD model when decision making was undertaken at the client's location. In the same figure, the simulation model ran three different groups of clients' requirements (as explained in section 6.2.3), presented on the x-axis. The mean duration of the entire customisation process in hours is presented on the y-axis: between 2.52 and 2.79 hours for all groups in the baseline model, while the same requirements take between only 1.13 and 1.99 hours under the CCRD model.

Table 6.1: Decision-making results of statistical comparison between baseline model and CCRD model for local decision making

Client group	Model	N[1]	Mean[2]	Std dev	Effect size	P value
Group 1	Baseline	1055	2.24	5.7	0.41	0.001 (P <0.05)
	CCRD	1055	0.40	0.32		
Group 2	Baseline	877	2.07	5.45	0.44	0.001 (P <0.05)
	CCRD	877	0.39	0.33		
Group 3	Baseline	337	2.08	5.53	0.38	0.001 (P <0.05)
	CCRD	337	0.38	0.37		

[1] Number of requirements [2] Mean decision time in hours

Table 6.2 displays the statistical values of the Wilcoxon non-parametric test, applied to test the significance of the difference between the output of baseline simulation model and the CCRD simulation model. In the same table the P value is <0.05 for all groups, which means that the null hypothesis, that is, that there is no difference between the baseline simulation model and CCRD model in the entire customisation process time, is rejected for all groups in this experiment, and the alternative hypothesis is accepted, that there is a significant difference at 95 per cent level of confidence.

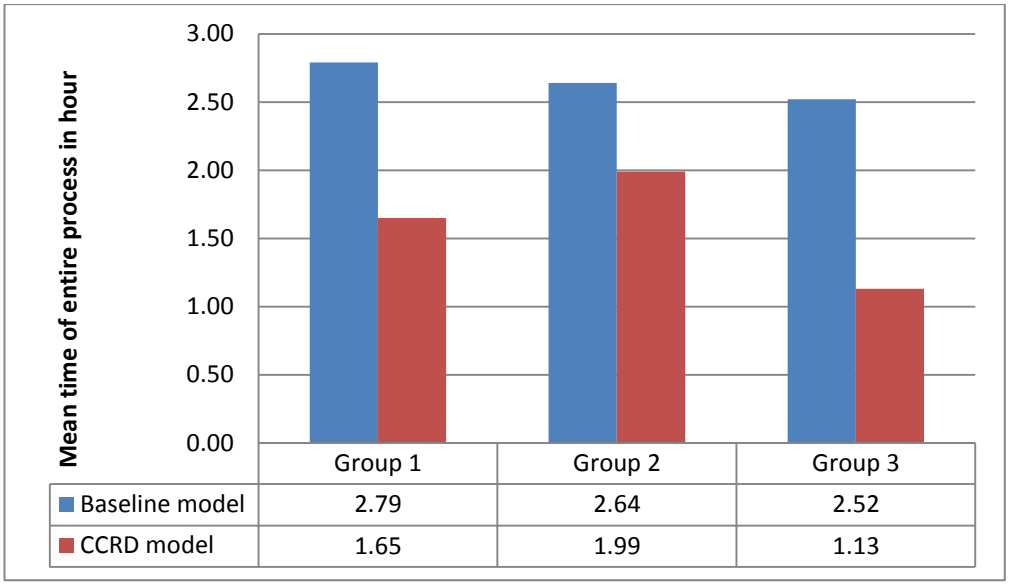


Figure 6.5: Mean duration of entire customisation process in baseline and CCRD models

In summary, this experiment was conducted to evaluate the CCRD model for local decision making by comparing the outputs of the baseline simulation model of three different groups containing 18 clients and 2479 requirements. It measured the impact of applying the CCRD

model to reduce the duration of decision making and the entire customisation process, as defined in the following hypotheses.

Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients.

Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients.

The first hypothesis (Hypothesis 1) was tested and confirmed as described in Results and analysis, section 6.2. Decision-making time was reduced significantly under the CCRD model compared to the baseline model. Also, the second hypothesis (Hypothesis 2) was tested and confirmed in the same way as Hypothesis 1, presented in section 6.2. The results are that the customisation process under the CCRD model is significantly shorter than under the baseline model.

Table 6.2: Entire customisation results of statistical comparison between baseline model and CCRD model for local decision making

Client group	Model	N[1]	Mean[2]	Std dev	Effect size	P value
Group1	Baseline	1058	2.79	5.69	0.15	0.001 (P <0.05)
	CCRD	1058	1.65	2.14		
Group2	Baseline	889	2.64	5.51	0.05	0.028 (P <0.05)
	CCRD	889	1.99	2.87		
Group3	Baseline	338	2.52	5.62	0.14	0.001 (P <0.05)
	CCRD	338	1.13	1.30		

[1] Number of requirements [2] Mean of decision time in hours

6.3 Experiment 2: Evaluation for CCRD model – Local development

The previous section presented the results of applying decision making locally through the CCRD model – local decision making scenario. This experiment aims to evaluate the second scenario of the CCRD model, which considers the concept of locality in decision making and some development processes in order to reduce challenges of communicating customisation requirements for multiple distributed clients. In this experiment, the hypothesis (Hypothesis 3) has been examined and confirmed. This refers to how the designed CCRD model with local decision making and local development processes for some clients' issues helps the

customisation process for distributed clients to be faster at decision making, in both the development process and the entire customisation process, than under the baseline model in the selected case study.

6.3.1 Experiment design and methodology

This section presents the experimental method and setting of the evaluation of the CCRD model for local development processes, explained in a conceptual model in Chapter 4. In fact, this experiment used the same settings as the previous evaluation experiments for CCRD models of local decision making (section 6.2). However, there was a single change in the model, namely the local development process at each client's location, as shown in Figure 6.8.

Local development process activity is at the clients' premises in order to deal with specific types of requirement, for example building reports based on clients' requests that neither change the system structure nor take much time from the central customisation team's dealings with requirements or critical requirements that need to be applied locally to avoid delays in communication across distributed boundaries. This experiment took critical requirements as an example and applied these locally, due to the historical data collected in a contextual review study (section 5.3) having no clear information about other types of requirements, since the CCRD model was designed to be compatible with various types of requirements. Local development activity had the same distribution of development process as the central team, but used only one resource at each location apart from the decision-making resource.

Regarding designing and running the experiment and avoiding repetition, section 6.2.1 explains the setting including simulation time and gives a description of each activity.

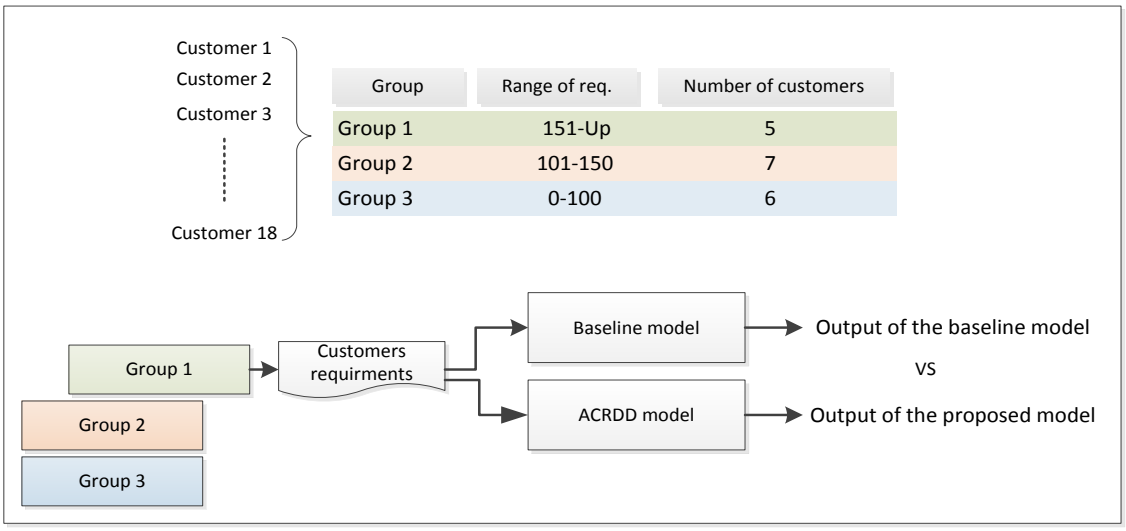


Figure 6.6: Sequence of experiment for different arrival data groups for local development

6.3.2 Results and analysis

This experiment measured the reduction in time achieved by applying the CCRD model's decision making time to the development and the entire customisation processes. In order to achieve this goal, the experiment input the same arrival data into the simulation model of the baseline model of customisation process and the simulation model of the CCRD model of local development process and compared the outputs to confirm or reject the defined hypothesis (Hypothesis 3).

6.3.3 Decision-making time

This experiment measured the reduction in time under the CCRD model for local decision making and the development process of clients' requirements. Figure 6.7 illustrates the mean duration of decision making for the baseline simulation model and the CCRD simulation model for local development. The x-axis in this figure represents the groups of clients and the y-axis the mean duration of decision making in hours. None of the three groups of clients' requirements, which ran one by one, had a mean duration for decision making of less than two hours under the baseline model; the same groups took less than a mean of 30 minutes decision-making time under the CCRD model. Furthermore, this experiment displays the statistical values of the Wilcoxon signed-rank test, applied to examine the difference between two models outputs. The mean durations of decision making under the baseline model for Groups 1, 2 and 3 were 2.24, 2.07 and 2.08 respectively, whereas under the CCRD model they were 0.39, 0.36 and 0.36, indicating a clear and significant difference between the models in terms of decision-making duration. The null hypothesis, defined as no significant difference between baseline model outputs and CCRD model outputs, is rejected and the alternative hypothesis accepted, that there is a significant difference in decision-making time in the CCRD simulation model for local development, because the P value for all three groups was 0.001, as shown in Table 6.3, and at less than 0.05 this is at the 95 per cent level of confidence.

6.3.4 Development process time

This experiment aimed to evaluate the CCRD model under the local development scenario that considers development at the client's location and performed by the distributed team, as explained in section 6.3.1. This section presents the reduction of time under the CCRD model for the local development of some requirements.

Table 6.3: Decision making results of statistical comparison between baseline model and CCRD model for local development process

Client group	Model	N[1]	Mean[2]	Std dev	Effect size	P value
Group 1	Baseline	1055	2.24	5.7	0.40	0.001(P <0.05)
	CCRD	1055	0.39	0.32		
Group 2	Baseline	877	2.07	5.45	0.41	0.001(P <0.05)
	CCRD	877	0.36	0.28		
Group 3	Baseline	337	2.08	5.53	0.33	0.001(P <0.05)
	CCRD	337	0.36	0.31		

[1] Number of requirements [2] Mean decision time in hours

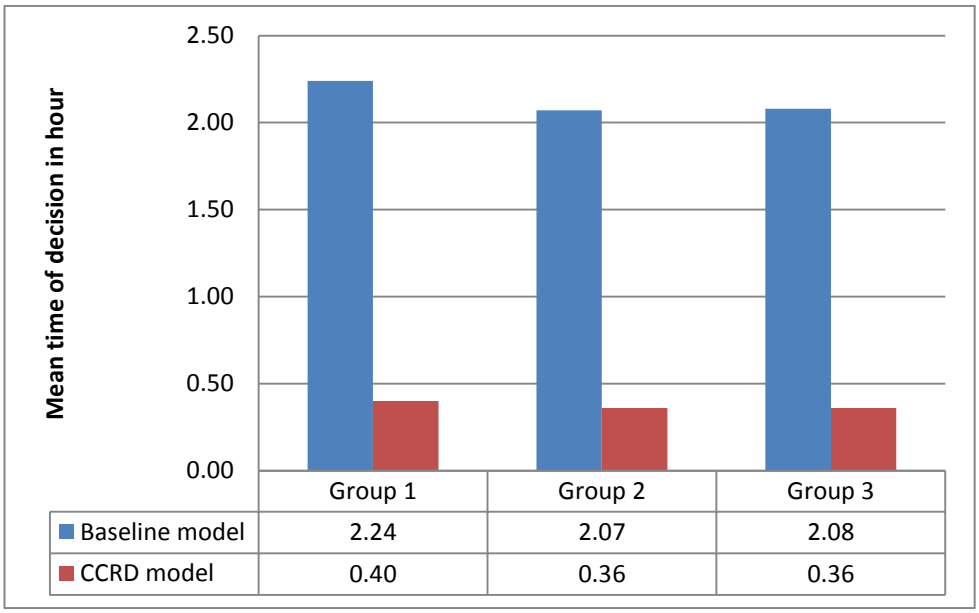


Figure 6.7: Mean duration of decision making in baseline and CCRD –- local development model

The three groups of requirements were input into both the baseline simulation model and the CCRD simulation model for local development. Figure 6.9 illustrates the mean time spent on development processes, including preliminary queueing time. From this it is clear that there is a significant difference between the mean duration of development in the baseline simulation model and the CCRD simulation model: instead of taking over 1.20 hours, the same requirements took on average between just 38 and 48 minutes.

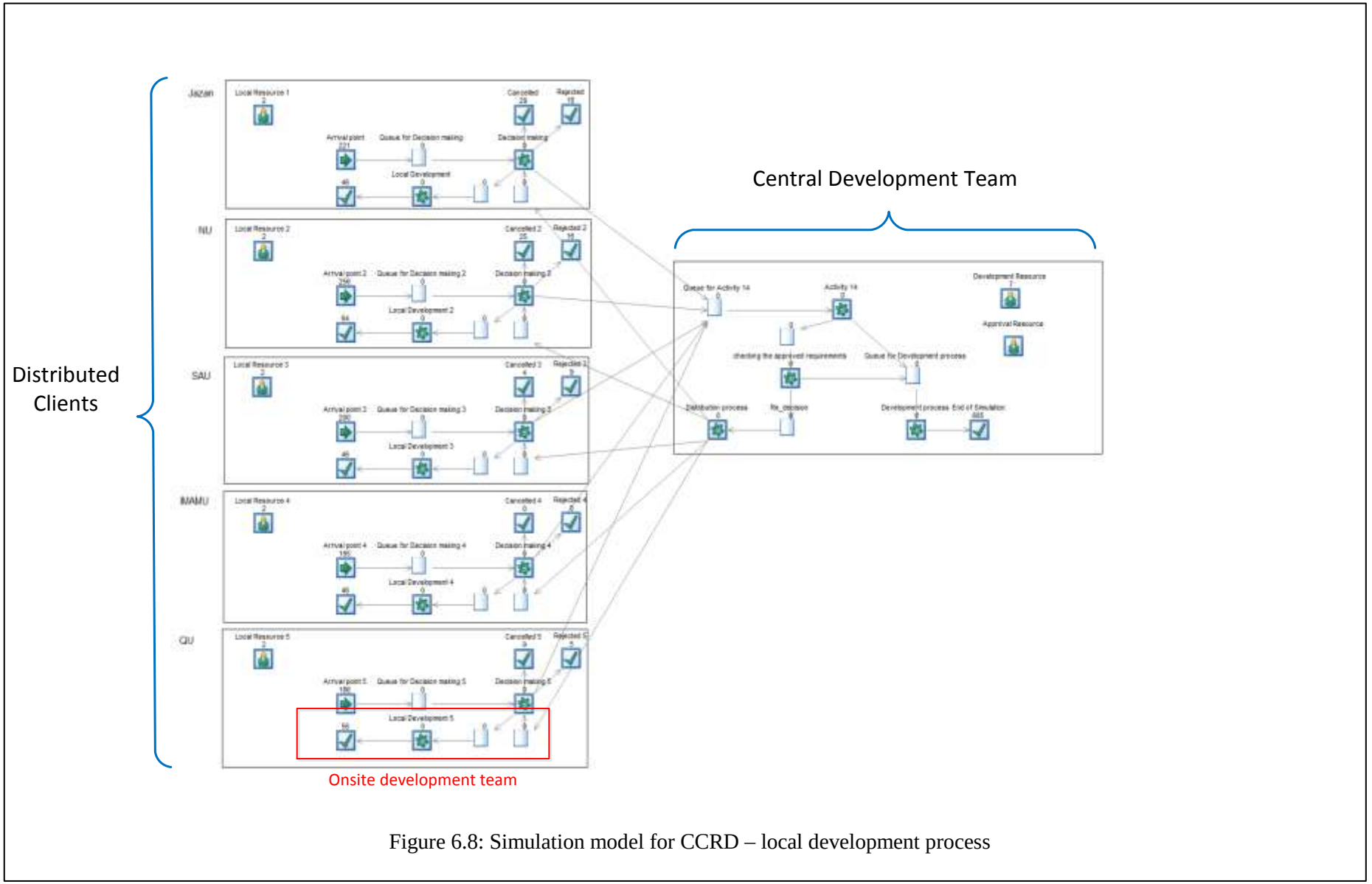


Figure 6.8: Simulation model for CCRD – local development process

In the same vein, the inferential data of statistical test shown in Table 6.4 refer to the significant difference between the outputs of the baseline simulation model and the CCRD simulation model using the Wilcoxon signed-rank test. The table shows that the development process in the baseline simulation model for Groups 1, 2 and 3 took 1.41, 1.34 and 1.22 respectively, on average. The same requirements were run under the CCRD simulation model, applying the local development process for some requests. The mean duration of the development process diminished to 48.6 minutes for Group 1, 46.8 for Group 2 and 37.8 for Group 3. In addition, Table 6.4 shows that *P* values of the all three groups are less than 0.05, considered significant in this experiment at the 95 per cent level of confidence. This means that the null hypothesis is rejected and the alternative hypothesis accepted, that there is a significant difference between outputs of baseline simulation model and CCRD simulation model in the duration of the development process.

6.3.5 Entire customisation process time

The duration of the customisation is the main concern for clients, being the time taken for their requirements to be completed and delivered. It is affected by any changes to the process. This experiment aims to examine the duration of the entire process in both the simulated baseline model and the CCRD simulation model for local development. Figure 6.10 illustrates clearly that the mean duration of the entire customisation process under CCRD for Group 1 and Group 2 is about the half that for the whole customisation process under the baseline simulation model for the same requirements, running at 1290 hours; that is, over 2.50 hours for both groups.

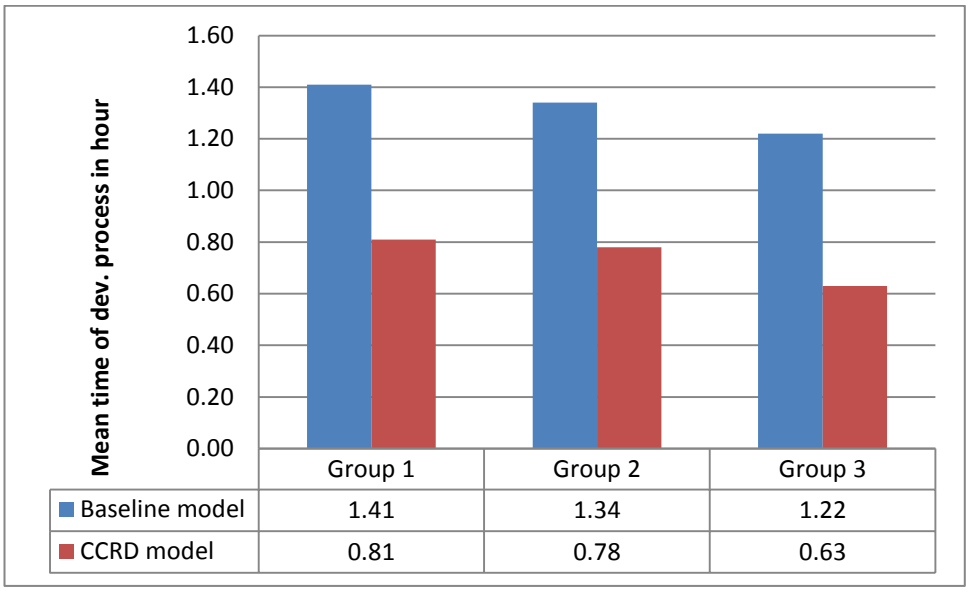


Figure 6.9: Mean duration of total development process under the baseline simulation model and the CCRD model – local development model

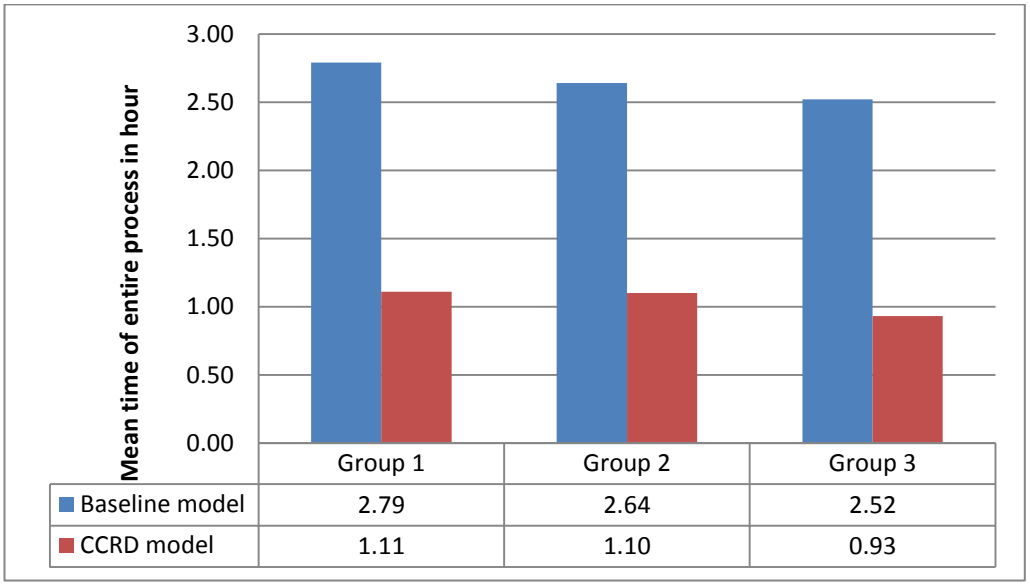
Table 6.4: Development process results of statistical comparison between baseline model and CCRD model for local development process

Client group	Model	N[1]	Mean[2]	Std dev	Effect size	P value
Group1	Baseline	943	1.41	2.24	0.16	0.001
	CCRD	943	0.81	1.28		(P <0.05)
Group2	Baseline	793	1.34	2.02	0.16	0.001
	CCRD	793	0.78	1.22		(P <0.05)
Group3	Baseline	297	1.22	2.04	0.16	0.001
	CCRD	297	0.63	1.00		(P <0.05)

[1] Number of requirements [2] Mean decision time in hours

Also, the mean duration of the whole customisation process for Group 3 shows a clear difference between baseline and CCRD simulation models, at 2.52 and 0.93 hours for the baseline and CCRD models respectively.

In terms of statistical analysis, the Wilcoxon signed-rank test was conducted and the results are shown in Table 6.5. In the table the mean duration of the entire customisation process of Group 1 was 2.79 hours under the baseline model, while the same requirements took 1.11 hours under the CCRD model. Also, the requirements of clients in Group 2 took 2.64 hours on average, falling to 1.10 hours under the CCRD model. The groups both show a significant difference in the whole duration of customisation process as the *P* value is < 0.05. Furthermore, this table shows for Group 3 the mean duration of the entire customisation process for clients' requirements. These requirements took on average 2.52 hours under the baseline simulation model and 0.93 hours under the CCRD simulation model. The *P* value of that group was less than 0.05, which means there is a significant difference between the duration of the customisation process in both models. Overall, the results of the Wilcoxon signed-rank test for all groups indicate that the outputs of the baseline simulation model and the CCRD simulation model show a significant difference in the duration of the entire customisation process, when *P* < 0.05 at 95 per cent level of confidence. That means the null hypothesis is rejected and the alternative hypothesis accepted; that there is a significant difference in the entire customisation process time between the baseline simulation model and the CCRD simulation model.



	Group 1	Group 2	Group 3
Baseline model	2.79	2.64	2.52
CCRD model	1.11	1.10	0.93

Figure 6.10: Mean duration of the entire customisation process under the baseline simulation model and the CCRD model – local development model

In summary, the evaluation of the CCRD model for local development was conducted in order to evaluate the impact of local decision making and various development processes (critical requirements, in this case) on the productivity of the customisation process for distributed projects. It was evaluated by comparing the outputs of baseline simulation model of three different groups containing 18 clients and 2479 requirements to measure the impact of applying local decision making and undertaking some development processes at client's own locations on decision-making time, total development process time and the entire customisation duration, as defined in the following hypothesis:

> *Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process.*

This hypothesis (Hypothesis 3) was tested and confirmed, as described and shown in the results and analysis section (6.3.2). Decision-making time was significantly shorter in the CCRD model than under the baseline model. Also, the development time is significantly different in the two tested models, which means that the CCRD model has a positive effect on the total duration of the development process, where some clients' requirements receive a quick decision and development.

Table 6.5: Entire customisation process results of statistical comparison between baseline model and CCRD model for local development process

Client group	Model	N[1]	Mean[2]	Std dev	Effect size	P value
Group 1	Baseline	1058	2.79	5.69	0.280	0.000
	CCRD	1058	1.11	1.29		(P <0.05)
Group 2	Baseline	889	2.64	5.51	0.241	0.000
	CCRD	889	1.10	1.24		(P <0.05)
Group 3	Baseline	338	2.52	5.62	0.213	0.000
	CCRD	338	0.93	1.02		(P <0.05)

[1] Number of requirements [2] Mean decision time in hours

The entire customisation process time shows a statistically significant difference in local decision making, overcoming the challenges of communication in customisation requirements across distributed boundaries by reducing delays in making decisions, applying development and completing the customisation process, then delivering the completed solution to clients' requirements.

6.4 Experiment 3: Findings of the confirmatory study

The objectives of the confirmatory study have been introduced (section 5.5), and confirm the outputs of the contextual review study (section 5.3), discuss the findings and explore the experts' view regarding locating decision making and some development processes at clients' premises. This section presents the findings of the semi-structured interviews (Figure 6.11) conducted with seven experts from the case study.

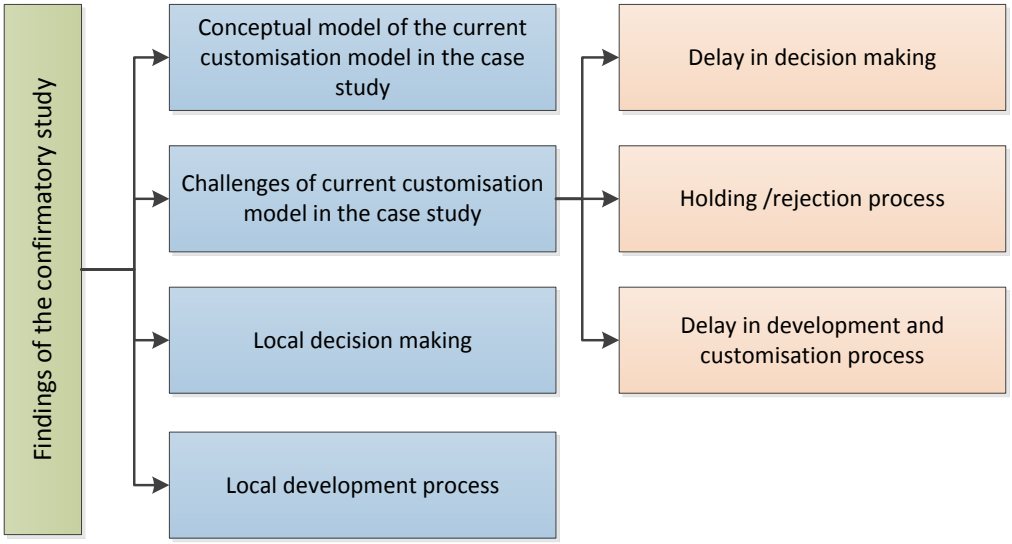


Figure 6.11: Structure of confirmatory study findings

6.4.1 Conceptual model of the real customisation process in the case study

The conceptual model design of the case study (Figure 5.3) was presented to interviewees to confirm it and to discuss some parts of that model such as the decision directions in the model. The model was confirmed by all interviewees; some stated that:

> *'The simulation reflects the current model of customisation in the company'.* (Int. 4)

and

> *'The simulation model reflects the actual case'.* (Int. 7)

In fact, the company introduced a slight change just before the confirmatory study, adding a new process into decision making for clients at some locations: the analysis of customisation requirements before sending them for decision making. However, the company does not have sufficient data to export since making that change, and two interviewees mentioned that it had caused only a minimal departure from the simulation, as stated here:

> *'The baseline model reflects 95% of the actual system and the difference lies in the new alteration, which comes under the role of the system analyst.'* (Int. 5)

and

> *'There is a new process added recently to the current model, which is analysing the request before making decisions, but does not yet apply in full, therefore there is no data to be collected from the system since introducing this process'.* (Int. 6)

6.4.2 Challenges of communication in customisation of the case study

During analysis of the collected data in the contextual review study, many issues were noted as implications of communication challenges in the actual case study. In the confirmatory study, the challenges were discussed with interviewees to confirm the findings and to explore their opinions of them. The following question was asked in the semi-structured interviews after pointing out to interviewees implications such as delays in decision making, and holding and rejection problems: 'What do you think are implications for central decision making?'

- **Delay in decision making**

One of the observed impacts of central decision making and distributed communications for customisation requirements is the delay in making decisions. This issue was confirmed as one of the customisation problems in the company; as one of the interviewees commented:

> *'The waiting time before decisions is real and has been noted in the company'.* (Int.3)

Another interviewee stated that:

> *'Decision making's distance from clients causes a long time between creating a request and its decision, while clients wait for solutions, thinking the request is already approved.'* (Int. 5)

Furthermore, in terms of the challenges to communication of clients' requirements in the distributed domain, one interviewee said:

> *'Yes, there is delay and misunderstanding between decision makers and client's locations. Also, challenges of communication across distributions locations do exist.'* (Int. 4)

There are many reasons behind the wait for a decision. During the semi-structured interviews, the interviewee emphasised variables that may cause the delay, starting with the human factor in discussing clients' requirements and communicating them to decision makers. This means that the experience and quality of the implementers at the clients' premises affect the duration of the decision-making episode, as follows.

> *'The human factor affects the waiting time before and during decision making. Sometimes the client's request is not clear to the decision maker and demands more information.'* (Int. 2)

> *'We supposed that there is no problem in communication between decision makers and implementers. However, some locations face problems in communication because the implementers are new or not professionals, either in the business or the system.'* (Int. 3)

> '*An issue we faced in decision making is where the request is sent by an unqualified implementer, which is unclear and causes issues like delays in decision making to understand the request, and more delay in further communication between client's location and decision maker to clarify that request*.' (Int. 5)

Another variable behind that delay is the timing and priority of the request, which mean that some requests have high priority and are picked to proceed before other requests, affecting the time in the queue. Some interviewees emphasised this issue as one of the more important factors, especially in dealing with multiple clients at the same time using the same resources – decision makers.

> '*The time varies, depending on the period of the year, based on the workload peak at the client's location and the priority of the request.*' (Int. 3)

> '*The decision-making queue is affected by the request's priority, the type of the request and time of the request.*' (Int. 7)

One of the problems with the delay in decision making is the clients' awareness and satisfaction, an issue mentioned by some interviewees with an implication for delays in communication and negotiation across distributed domain of the client's requirements. As the client sends the request and waits for a solution, the client may realise that the request is unclear and thus time is spent in a holding process to communicate with either this client or the representative of the company at client's premises to understand the request and make a decision, as stated by some interviewees.

> '*There is a problem in the current system, which is that clients complain at the delay in decisions and resolving the request as they don't know about the requests of other clients and the queue in front of their request*.' (Int. 7)

> '*The distance of decision making from the client causes a long time between creating the request and the decision, while clients wait for a solution, believing that the request has been approved.'* (Int. 5)

Also, the latter interviewee stated that:

> *'The time before decision making is correct: it takes a long time and the client and implementers notice this more than people at central sites.'* (Int. 5)

- **Holding and rejection process**

In the actual case model of the customisation process at the selected company, making decisions on clients' requirements takes different forms, as mentioned in section 5.4.5. The holding process is one of these decisions, when the request is unclear and the decision maker needs further explanation to understand it. The main purpose for this holding process is that unclear requests arrive at the central decision-making section that need to be discussed with clients. This issue was observed in the contextual review and in this study, as well as being mentioned in the following answer of one interviewee:

> '*We used the holding process only for the unclear requests which need more investigation. Some requests are difficult to give decisions on.'* (Int. 1)

Regarding the holding process, other interviewees in a company department called ERP said that:

> *'In the ERP system, we put on hold any request that takes more than a week in discussion. Sometimes the request conflicts with other systems and it takes time to solve that conflict.'* (Int. 2)

> '*Sometimes the issues in the holding process are cancelled if they take a long time.'* (Int. 6)

> *'It takes a long time in discussion to understand some requests.*' (Int. 3)

The rejection process happens if the request is turned down, to be approved only after discussion with a client. This also causes a delay in decision making as the communication with clients has to take place across the distributed domain and involves discussion between decision making and rejection sections:

> *'The rejection process happens if the request is accepted in general but the needs aren't clear and clarified in the rejection process and then approved or rejected.'* (Int. 6)

Another interviewee indicated the challenge of communicating with clients directly over distributed boundaries, as he mentioned:

> *'The communications are with our implementation in clients' locations. We had an experience with one client who hasn't an implementer at his location. We stopped dealing with them after a week due to they don't understand the business and the system as well.'* (Int. 1)

In fact, the holding/rejection process and communication with clients for explanations is a problem behind many issues such as client satisfaction and delays in resolving requirements. This is discussed and mentioned during the semi-structured interviews, as one of the interviewees said:

> *'The distance of decision making from clients causes a long gap between creating the request and the decision, while the client waits for the solution thinking that the request had been approved.'* (Int. 5)

- **Development and entire customisation process**

In the selected case study the process of customisation is a sequence of sub-processes: collecting requirements; making a decision; developing the request; and delivering the solution. Therefore, a delay in one of these sub-processes will affect the rest. In this case, any delay in the decision-making process, either in the queue before making a decision or in the holding or rejection process after a decision, may affect the total development time and thus the entire customisation process. One interviewee referred to the impact of a delay on the time of delivery of solutions:

> *'Tasks take a long time between decision making and starting development. We talk to project managers many times to make changes to reduce that time, as it affects the delivery time and annoys clients.'* (Int. 5)

Another interviewee said that:

> '*Some requests need only half an hour, however most time is spent in queuing.*'(Int. 1)

Further interviewees mentioned the impact of changing a request's priority and discussing this with clients throughout the holding and rejection process:

> *'Sometimes we use the holding process in the wrong way which prolongs the total duration of the customisation process.'* (Int. 1)

> '*There is a reason for that delay, which is the difference in priorities and types of requests.'* (Int. 3)

In addition, interviewees stated that:

> *'The development process takes time, depending on the priority of the tasks and the person who assigns the priority is the project manager – and the client isn't aware of this process.'* (Int. 5)

> '*The delay in development happens in the actual system and the reason is the changing priorities based on the timing of requests. For example, if a client requests something relating to the beginning of a season and the season has passed before the request is completed, the priority of that request changes to low and it will be completed when the developers have time to do it.'* (Int. 7)

6.4.3 Impact of local decision making on decision time and total time of entire customisation process

This research investigates the concept of locality in the distributed domain. To that end, this section of the confirmatory study discusses the impact of making decisions at the client's location on the decision-making time and the total time taken for the customisation process, through semi-structured interviews at the selected case study, as explained in Chapter 5. Collecting clients' requirements in software development is an important part of the development life cycle. During collection there is discussion and negotiation to make requests clear and understandable for decision makers and developers. In distributed software projects, communicating requirements across distributed boundaries is a problem, as mentioned in Chapter 4.

In this study, the participants confirmed and emphasised the importance of discussing the client's requirements at the client's location, as the following interviewee stated:

> '*We had an experience with a one client who hasn't an implementer at his location. We stopped dealing with them after a week due to them not understanding the business or the system either.'* (Int. 1)

Some interviewees emphasised the benefits of taking decisions at the clients' locations as this made negotiations on customisation requirements easier. Also, the requirements received by the central team would be clear and agreed by both the decision maker and the client, so would not need more communication about the holding and rejection process, unlike the situation with central decision making, as the following interviewees mentioned:

> *'I expect that if the decision were made at the client's location it would reduce the time for decision and the entire process.'* (Int. 1)

> *'I agree, moving decisions from the central to the client's location would reduce the challenges of communication and the time needed for discussions and the waiting time in the customisation process.'* (Int. 3)

Another interviewee stated that:

> *'Making decisions at the client's location would reduce the time and challenges of communication between client, decision maker and implementer.'* (Int. 4)

Furthermore, the experts referred to the satisfaction of the client if decisions are made locally with face-to-face discussion, which increases the awareness of clients about the priority of their request and other procedures. As there interviewees stated:

> *'Locating the decision making at the client's location will make a difference to decision and negotiation time, and to the satisfaction of the client as he discusses the requests and priorities directly with a decision maker and is aware about that.'* (Int. 5)

> *'If each client has a decision maker it will affect the decision time and customisation time. Making decisions at a client's location has a problem. They lack the awareness of what is happening at other clients' locations when they all use the same version of the system.'* (Int. 7)

In addition, experts agreed that local decision making would eliminate the holding and rejection process, as the required communication would take place with local decisions made face-to-face with clients at their locations:

> *'Indeed, this change in decision making location would eliminate the holding and rejection processes and all waiting time will be per site.'* (Int. 6)

Although local decision making would make a significant difference to the customisation process in the distributed domain, there are some concerns and challenges that might affect the change in location of decision making. During the confirmatory interviews, concerns were

mentioned by many interviewees: the importance of placing a qualified and experienced decision maker in that role at a clients' location in a professional manner.

> *'The experience of the implementer is very important, as sometimes clients request something that affects other parts of the system, so at this time the implementer explain that to him.'* (Int. 5)

> *'If the decision maker is qualified, seniority is clear during communication and discussion. A very important point should be considered, which is the person who make a decision should be qualified to do that, if he stays at the client's location, and should report the client's requirements in very clear way to be understood by the developers.'* (Int. 6)

Another concern regarding the change of decision making location from the central team to the premises of clients is the differentiation in decision making. One of the interviewees said:

> *'There is a problem in making decisions at client's locations, the differentiation in decisions. Now, the central decision is made at the same level.'* (Int. 1)

6.4.4 Impact of local development process on decision time, development process and total time of entire customisation process

Local development is one of the main aspects that has been discussed in the quantitative section in this research through simulation, and in this section through a qualitative approach with semi-structured interviews. In this confirmatory study, the CCRD model for local development was presented to the interviewees in order to obtain their opinions on this change and how much that would help to reduce the challenge of communicating customisation requirements across distributed boundaries. In general, the interviewees agreed that some development at the client's location would help to reduce the challenge of communication challenges in the distributed domain. It could cut the time for the entire customisation process, besides increasing client satisfaction. As some of them stated:

> *'No doubt if we apply some development at the client's location it would save time waiting and processing. Also, it will solve some challenges of communication between implementers and decision makers.'* (Int. 3)

One of the interviewees, who was working as an implementer at a client's location, said that:

> *'This question touched the needs of implementers and clients as well. I believe that if one developer is located at a client's location it would make a huge difference in customisation process in both time and effort.'* (Int. 5)

The company used to use local decision making and development processes at one of their clients' locations and it was a successful experience. It was faster and better at understanding and resolving clients' requirements in face-to-face discussions. As one of the interviewees said:

> *'We had a project with a decision maker and a developer in the client's location and the response and the communications was better than now.' (Int. 4)*

However, a couple of concerns were raised by the interviewees as challenges might be faced if some development processes are applied at clients' locations. These are the problem of using a single-system version for multiple clients, and the motivation of different teams and the difficulty in deciding which requirements to send to the central team and which to apply locally:

> *'The difficulties would be in the decision on where this task should be, locally or in the central team.'* (Int. 1)

> *'Agreed: local development, keeping the single version, will be better and faster and will affect the whole customisation process.'* (Int. 2)

Another interviewee gave the example of some requirements taking a long time in queues before starting the development process, which would be straightforward with local development process:

> *'Yes, there are requests that need only a short time in development. However, it takes a long time in queues before the development process. Moving development process for some requests would reduce the development time but it will affect the single version.'* (Int. 6)

However, systems that do not use the single version for different clients will benefit by locating a small development team at clients' premises, as one of the interviewees indicated that the web system in the company uses the same customisation process model. However, there is no single version for all clients as in other systems in the company, which would be useful to be applied like the CCRD model for local development, as in previous cases.

> *'In the web application there is no single version. So, the proposed model will be better in time with fewer challenges in communications.'* (Int. 4)

On the other hand it was agreed by all interviewees that the problem of a single version was the difficulty in deciding whether a requirement should be sent to the central team or applied locally. Clients require new reports, or amendments and sometimes fixes to problems in others, which takes a long time in the customisation process, from decision making through to development. This type of requirement could applied at clients' locations, which surely would reduce decision making, development and entire customisation times, yet not affect the single version of the system, as the following interviewees said:

> *'There is a suggestion, apply the reports development in clients' locations which will reduce the time of waiting and development. Also, it will reduce the communication challenges in the distributed domain. Since, 3 of 10 requests are reports which take time and there is no problem to make it locally.'* (Int. 3)

> *'I believe that if one developer were located at client's location it would make a huge difference in customisation process in time and effort. Especially in case of reports.'* (Int. 5)

> *'Moving development process for some request would reduce the development time but it will affect the single version. The correct changes should be locating the reports development at client's location.'* (Int. 6)

The suggestion of local development had been used before in a project, as one interviewee said:

> *'Local development is a good idea to reduce the time of development and entire customisation process; we used this with one client, we located a developer to complete his reports requests and that was very useful and faster.'* (Int. 7)

To summarise, the confirmatory study was designed to present the findings of the contextual review (section 5.3) to confirm the findings, to discuss making decisions and some development process at clients' locations (CCRD model for local decision and local development process) and to present the initial results of the evaluation on the simulation of the case study model and CCRD model, in both scenarios. Semi-structured interviews formed the method for conducting this study, which was with seven experts in the selected company, as explained in section 5.5. During the interviews, four aspects were discussed.

I. The conceptual model of the system in the company, simulated as the baseline of the evaluation process in this research (section 5.4.5). The conceptual model was confirmed to reflect the actual system in the selected case study.
II. The challenges of the communication customisation requirements across distributed boundaries in the case study: delays to decision making, development and the entire customisation process, also, the time spent by unclear requests in the holding and rejection process. These challenges were agreed by interviewees to be prominent issues for practitioners and clients, and to affect the productivity of the customisation process in the case study domain.
III. Local decision making at clients' locations was discussed during interviews and was agreed to be a solution by eliminating the holding and rejection process and reducing decision-making time by enhancing face-to-face negotiation at the clients' premises. There were some concerns, but changes such as providing qualified decision makers for each location were expected to reduce delays in the customisation process and increase the satisfaction of clients, as they would be more involved in decisions.
IV. Applying some development processes at clients' locations was discussed and it was agreed by interviewees that a local development process would reduce the challenges of communication across the distributed domain and improve productivity, as well as client satisfaction. However, it was felt that it could cause another issue: breaking up the single

version of the provided system in that case study. Consequently, they encouraged applying local development to reports requirements only in systems using a single version.

6.5 Chapter Summary

This chapter presented the evaluation of this research, including the settings used, the design of evaluation experiments and the results obtained in each experiment. The evaluation process went through three experiments. The first two were for evaluating the CCRD model with local decision making and local development processes at clients' locations, using a simulation method to compare the baseline simulation model of the selected case study and the simulation model for each scenario of the CCRD models. The data collected from the selected company in the contextual review study were arrival requirements for both simulation models in each experiment, approximately 2479 customisation requirements for 18 distributed clients. The results obtained by these experiments answered the following research questions.

RQ2: What is the impact of making decisions locally on the total decision period and the total customisation period of software customisation processes in the distributed domain?

RQ3: How much would undertaking the process of development of minor tasks at clients' locations reduce the total development and customisation period?

Also, they confirmed the following hypotheses.

Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients.

Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients.

Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process.

The results show that using local decision making at clients' locations reduces the duration of decision making and the entire customisation process. Also, allocating a development team to a client's location for some development processes reduces the duration of the development process as well as the entire customisation process. The third experiment, conducted at the case study company as explained in section 5.3.1, used semi-structured interviews in order to

confirm the baseline model and the findings of the contextual review study and to obtain the experts' opinion on the challenges of current model of the company, explained in Chapter 5. It investigated the impact of using local decision making and local development processes on the customisation process in the distributed domain.

This experiment had seven experts from different departments in the company and the findings show that the design of their working model confirmed that the challenges include delays in the customisation process and prolonged distributed communications in some processes. All were agreed by the interviewees to be prominent challenges in their work. Furthermore, the concept of locality was discussed with them through local decision making and the development process. They agreed that local decision making would enhance the productivity of decision making and the entire customisation process. Also, a local development process would help to reduce the duration of decision making, the development process and the entire customising process, and to overcome the challenges of communication across the distributed domain (see sections 6.2 and 6.3). Moreover, suggestions and concerns were discussed through this experiment, as shown in section 6.4, by two different methods: the simulation method to test the validity of the CCRD model, and ascertaining to what extent this would confirm the defined hypothesis and answer the questions of this research.

Chapter 7: Discussion and Research Findings

7.1 Introduction

This chapter discusses the results presented in Chapter 3 to identify the challenges and factors in customising software products across organisational and distributed boundaries, in order to answer Research Question 1. In addition, it discusses the results presented in Chapter 6 to evaluate the CCRD model of local decision making and local development in communicating the customisation requirements for distributed clients. This is supported by the findings of the confirmatory study presented in section 6.4, to answer Research Questions 2 and 3, and the three hypotheses. At the end of this chapter there is a section on the limitations of this research.

7.2 Challenges of Customisation of Software Development in the Distributed Domain

This section discusses the challenges identified by a review of the literature on topics relating to the customisation of software products, for instance local practices such as the Agile approach in the distributed domain (Chapter 2). These challenges are presented in the form of a framework termed the FCCSD (section 3.2). The FCCSD identifies a number of challenges and factors, categorised into four main categories based on the type of challenge and its impact, as explained in section 3.2 and shown in Table 7.1.

Table 7.1: List of challenges addressed by the FCCSD

Communication	**Project management**	**Knowledge management**	**Configuration and integration management**
• Time zone • Trust • People • Cultural difference • Collaboration	• Risk management • Process management • Quality • Planning	• Documentation • Communicating requirements • Group awareness • Tools and technologies	• Integration • User acceptance test

In addition, this section discusses the findings of the preliminary study conducted to confirm the FCCSD components. This was undertaken with 19 experts in the area of software development (section 3.3) and identified challenges for the FCCSD, answering the following research question:

RQ1: What are the challenges identified in the literature in customising software products across organisational boundaries in the distributed domain?

The following subsections discuss the content of the FCCSD and the results of the preliminary confirmatory study for each component (section 3.2).

7.2.1 Communication

Communication is one of the main challenges for customisation projects in the distributed domain, as the literature suggests (section 2.5), and confirmed by the preliminary study. The relevant section of the FCCSD shown in Figure 3.1 and listed in Table 7.1 shows that four factors impact on communication in the distributed domain when local practices are applied during the customisation process, such as decision making and development: Time zone; Trust; People; and Cultural difference and collaboration. Those factors revealed statistically significant participant agreement in the preliminary study as they influence communications in the distributed domain and multiple team development projects, as shown in Figure 3.7 and Figure 3.8, and explained in section 3.2.3. However, the presence of these factors relies on the number of development teams in distributed projects. For example, the time zone issue occurs in projects operating across different countries, termed GSD projects, when there are different time zones or the different organisations have different working hours. However, this factor can be a benefit rather than a challenge as it enables software to be built 24 hours a day, as is the practice in several western companies that outsource to eastern countries, exploiting the difference in time zones (Gupta, 2009). In the same way, cultural difference is a feature of projects where team members speak different languages or are from different cultures, making understanding and communication more difficult. This invariably occurs in GSD and in outsourced projects (Krishna, Sahay & Walsham, 2006).

In summary, communication in general is the chief challenge in distributed projects and is a feature of customisation projects for multiple clients using local practices.

7.2.2 Project management

Project management is an essential aspect of any software project, dealing with the planning, monitoring and control of all aspects of development projects (Atkinson 1999). Software engineering in both development and customisation requires a high level of management in order to produce high quality software products promptly to stockholders' requirements, achieving customer satisfaction (Cohen et al., 2004; Bjarnason, Wnuk & Regnell, 2011). The FCCSD project management component (Table 7.1), apart from the technical factors, provides

various management factors that are important to handle customisation projects in the distributed domain, such as risk management, process management, quality and planning. These are indicated as challenges and impact on the development process in the distributed domain as well as in Agile development projects. Moreover, the project management component and related factors revealed statistically significant results in the preliminary study, as shown in Table 3.9 and Table 3.10. These results indicate the importance of project management in the customisation process and the impact of risk management, process management, quality and planning on customisation projects, playing a vital role in a project's success.

As project management is assumed to be a key topic of Agile development methodology, Agile approaches have come to manage a client's requirements, documents and people in the software development process, as indicated in the Agile Manifesto (Kent Beck et al. 2001). Also, distributed projects with multiple clients require a high management input (da Silva, França & Prikladnicki, 2010). As it is such a critical aspect of a software project, the FCCSD has a component for project management. There are four factors impacting on the project management of customisation projects for multiple clients in the distributed domain, namely risk management, process management, quality and planning.

7.2.3 Knowledge management

In software development projects there is a significant amount of knowledge exchange between teams (Bowen & Maurer 2002), and dealing with this volume of information plays an important role. The customisation of software products for multiple clients requires managing their requirements, thus knowledge management features in the FCCSD and has four factors for distributed customisation projects, namely documentation, communicating requirements, group awareness, and tools and technologies. In any customisation project information is a critical element in its success or failure. The following sections discuss each factor and its impact on customisation projects.

- **Documentation**: this represents a customisation challenge, especially among distributed development teams. The Agile development approach emphasises communicating informally about requirements among team members by face-to-face contact (Kent Beck et al. 2001), but this is not applicable to distributed projects. Documentation plays a vital part in managing the absence of face-to-face and informal communication (Motira & Herbsleb, 2001). Furthermore, results of the preliminary study emphasise that documentation is a knowledge management challenge to the customisation process, as revealed by the statistically significant results in Table 3.12. However, success in customisation projects for multiple clients in the distributed domain demands a balance in the level of documentation

to overcome trust problems (Ali Babar, Verner & Nguyen, 2007) in distributed communication, also to keep the whole process light and rapid, as emphasised in the Agile Manifesto (Kent Beck et al. 2001).

- **Communicating requirements:** the communication of customisation requirements is a critical practice in customisation projects, especially for multiple clients in the distributed domain, as indicated in the literature (section 3.2.5) and confirmed in the preliminary study, where communicating requirements revealed statistically significant agreement among the 19 experts, as shown in Table 3.12. Any misunderstanding of clients' requirements will impact on the whole project in terms of cost, time and effort as well as stakeholder satisfaction. In addition, communicating requirements needs special care as it involves interacting with clients who have different levels of knowledge and experience, besides different cultures. Moreover, the communication is among development team members located at different venues. All this communication about clients' requirements for the customisation process gives this factor a high level of importance in the FCCSD compared to other knowledge management components. Therefore, this study focuses on communicating customisation requirements and their impact on the whole process by investigating local decision making and development practices to reduce the challenges in the distributed domain, as presented in Chapters 4, 5 and 6. In addition, sections 7.4 and 7.5 of this chapter discuss communicating customisation requirements for multiple clients.
- **Group awareness:** the distribution of the software development team members results in a lack of awareness among those involved in customisation projects that may cause misunderstandings of clients' requirements or involve further communication among the team. Agile software development and its locality concept emphasise interaction between clients and their development teams (Kent Beck et al. 2001), and group awareness is one of the main factors in software customisation projects in the distributed domain, as identified from the literature (section 3.2.5) and confirmed by experts in the preliminary study. This is shown in Table 3.12, revealing the statistically significant result that group awareness is a knowledge management factor in customisation projects. The implications of enhanced awareness among teams and clients involved in customisation projects may include a reduction in the number and duration of further communications to uncover information and may prevent conflicts with existing code or important functions (Herbsleb, Grinter & Finholt, 2001; Fogelström, 2010).
- **Tools and technologies:** tools and technologies pay a vital role in distributed development, reducing the challenges imposed by distance to provide a communicative environment for the involved teams. In the literature, much research has investigated the importance of technologies in distributed projects (section 2.2.4), thus the FCCSD includes this factor as a

knowledge management component since it impacts on communicating requirements among teams and clients, helps to improve group awareness among involved teams and supports the exchange of documents over distributed and organisational boundaries. Table 3.12 shows a statistically significant result for tools and technologies in the preliminary study, indicating that e-mails, video conferences and share points (Table 2.1) are important in customisation projects in the distributed domain. Also, such tools play a critical role in easing communications among teams and clients in customisation projects. However, there are many challenges involved with these tools, as the literature indicates, and they cannot match face-to-face contact. So, the CCRD model presented in Chapter 4 and discussed in sections 7.4 and 7.5 relies on face-to-face contact rather than technology for critical practices such as making decisions about clients' requirements.

7.2.4 Configuration management

Software development projects conducted in the distributed domain will at some stage face challenges in integrating the teams and the software, especially in projects across multiple teams and organisations (del Nuevo, Piattini & Pino, 2011). Therefore, one of the main components in the FCCSD is configuration and integration management, emphasising the importance of integration and coordination among teams engaged in distributed customisation projects. Two factors impact on configuration and integration management in customisation projects across distributed boundaries.

- **Integration:** in the process of customising software products across distributed and organisational boundaries there will be multiple versions to meet clients' incremental change requirements or new requirements (Bosch & Bosch-Sijtsema 2010). Therefore, the integration process is emphasised to ensure a new version is compatible with the current version, or a client's version suits the organisational system. Emphasis is also laid on using the version control concept and technology to work with and move smoothly from version to version in the customisation process across customer boundaries. The results of the preliminary study (section 3.3) indicate a statistically significant result for integration, as shown in Table 3.14, which means that the integration process is one of the most important factors of configuration management, impacting on customising projects in the distributed domain. So, integration should be considered by customisation teams in order to develop and deliver software that is compatible with existing systems and fulfils clients' requirements to their satisfaction.
- **User acceptance test:** most software testing in the software development process, such as integration tests and unit tests, is performed in development time by the development team

(Gopalakrishnan, 1996). However, user acceptance tests require clients to be involved to ensure that the software meets all their requirements. Therefore, these tests are included in the FCCSD as key factors of configuration management, aiming to address testing across organisational boundaries. Since the FCCSD aims to identify the challenges of customising software across organisational and distributed boundaries rather than within development teams during the development life cycle or Agile development sprint, the user acceptance test has been selected as one of the challenges in that domain (Sengupta, Chandra & Sinha, 2006). Moreover, the preliminary confirmatory study (section 3.3) shows a statistically significant result for user acceptance testing as an important factor in the customisation process across organisation and distributed boundaries, as shown in Table 3.14. Also, it impacts on the integration component, emphasising the integration between users and development teams to deliver appropriate software that meets clients' requirements.

In summary, section 7.2 presents a discussion of the challenges and factors of the process of customising software products in projects conducted over organisational and distributed boundaries. These challenges are identified in the literature and presented in the FCCSD, then confirmed by 19 experts in software development projects in the confirmatory study in section 3.3 While the literature review covers frameworks discussing the challenges of distributed projects during the development process, or collocated projects such as those discussing using Agile approaches in distributed or collocated domains (section 2.2 and 2.4), the FCCSD fills that gap and provides a framework for the customisation process across organisational and distributed boundaries and answers Research Question 1:

> *RQ1: What are the challenges identified in the literature in customising software products across organisational boundaries in the distributed domain?*

Therefore, the FCCSD provides a list of challenges and factors that need more consideration during the process of customisation (Table 7.1), and informs project managers, practitioners and clients of the key components of customisation projects, including management and technical aspects that demand further attention.

7.3 Challenges of Communicating Customisation Requirements

The previous section discusses the challenges and main factors of customisation of software products across distributed boundaries. One of the challenges is that of communicating customisation requirements, since this demands special care as it involves interacting with clients who have different levels of knowledge and experience, besides different cultures in some cases. In addition, all customisation activities rely on client requirements. Therefore,

communicating client's requirements among development team members located at different venues and client's locations gives this factor a high level of importance in the FCCSD compared to other factors. Also, any misunderstanding of clients' requirements will impact on the whole project in terms of cost, time and effort as well as stakeholder satisfaction.

Therefore, this study focuses on communicating customisation requirements and their impact on the whole process by investigating local decision making and development practices to reduce the challenges in the distributed domain. The following sections discuss the impact of local decision and local development practices on customisation process and reducing the challenge of communicating clients' customisation requirements in the distributed domain.

7.4 Local Decision Making in Customisation Projects

Through the CCRD model, local decision making in the course of customisation projects has been discussed as a way to reduce the challenges of communication in the course of customisation requirements in the distributed domain (as shown in Figure 4.1). These take the form of delay in achieving decisions and a long drawn out customisation process for projects with multiple distributed customers. Evaluation experiments for the CCRD model for local decision making in customisation projects (presented in section 6.2) aimed to examine the following hypotheses.

> *Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients.*

> *Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients.*

The following sections discuss the findings that support Hypothesis 1 and Hypothesis 2 and their statistical significance.

7.4.1 Impact of local decision on decision time

The results of the experiment show a significant reduction in the duration of decision making for customisation requirements when decisions are made at a client's location, as shown in Table 6.1. This experiment involved three different groups of customers with different numbers of requirements (0–50, 51–100 and 101–upwards) over 1290 hours (the observation period) (see Figure 6.3), and all show a significant reduction in the time taken to achieve a decision on

customisation requirements. According to the CCRD model, the reason is close contact between client and decision maker; any distance causes delays when unclear customisation requirements are communicated across distributed boundaries to understand the request, with further delays to clarify that requirement (Int. 5 and Int. 2 in section 6.4.2). Another issue is the practice of dealing with requests on the basis of their assigned type and priority level, especially when simultaneously dealing with multiple clients using the same resource – decision makers (Int. 7 and Int. 3).

There are three advantages of allocating decision making to a client's premises in customisation projects:

1. It increases clients' awareness and satisfaction, as it highlights the implications of delays in the communication and negotiation of requirements across the distributed domain. When clients send a request out and wait for a solution they may well realise that it is unclear. Time is thus lost in a holding process while communicating with either the client or the company's representative on the premises to understand the request and make a decision (Int. 7). With local decision making, clients are involved in requirement negotiation and prioritisation, reducing the length of time taken for any clarification or discussion of priorities.
2. It obviates the need for the holding process where unclear requirements are held for discussions involving contacting the client location (Int. 1 in section 6.4.2). One interviewee indicated that sometimes a request conflicts with other systems and takes time to resolve, slowing down decision making (Int. 2). Sometimes a requirement in the holding process is cancelled only when much time has already been wasted in discussion just to understand it (Int. 3). Therefore, eliminating the holding process by negotiating clients' requirements on the premises helps to shorten the process.
3. It assists the definitive rejection decision, as explained in section 6.2.1. In the CCRD model, making decisions locally helps to decide on a definitive rejection after discussion with clients, shortening the decision-making period. Otherwise a rejection decision must wait for discussion with clients before re-approval or rejection, which happens when a request is accepted in principle but requires further discussion at the client's location to clarify details (Int. 6).

In summary, making a decision on customisation requirements at a client's location in the distributed domain reduces decision-making time and is a statistically significant result of Experiment 1 in section 6.2.2, supported by study findings. This confirms Hypothesis 1 and answers Research Question 2.

7.4.2 Impact of local decision on total duration of customisation process

Changing any activity in the customisation process, from collecting clients' requirements through to delivering the solutions, will affect the duration of the entire process. Therefore, the evaluation experiment (section 6.2) examined the impact on the entire customisation duration of making decisions at the client's premises. Results of the experiment are presented in section 6.2.4. This experiment was undertaken with three separate groups with different numbers of requirements (0–50, 51–100 and 101 upwards) observed over 1290 hours, as shown in Figure 6.3. All show a significant reduction in the length of the customisation process when decisions are made at a client's location (Figure 6.5 and Table 6.2). These results indicate the impact of making local decisions on a customer's requirements on the total duration, from collecting requirements to delivering solutions including development, testing and verification. Besides the reduction in time, when the CCRD model is applied with local decision making there are the following benefits.

1. Clear and agreed requirements are communicated between clients and decision makers, reducing the challenges of the distribution domain and consequent misunderstandings and wrong priorities that waste time in further communication, and in the holding and rejection queues (Int. 1 and Int. 3 in section 6.4.3).
2. Using requirements that have been agreed by clients and decision makers reduces the frequency of messages between customisation team members such as client, decision maker and developer, cutting the total duration of the customisation process (Int. 4).
3. Locating decision making at a client's premises and involving the client in decisions over requirements make a significant different to negotiation time and client satisfaction, enhancing the client's awareness of the priority of a requirement and its expected end time (Int. 5 and Int. 7).

To summarise, making decisions at a clients' location in the distributed domain has many advantages, reducing the duration of the customisation process and improving satisfaction with the resolution of requirements. However, participants in the confirmatory study indicate some concerns with local decision making.

1. The person to be located at the client's premises must be a qualified and experienced decision maker with a professional manner (Int. 5 and Int. 6) to reduce the challenge of understanding the client and to foster an environment conducive to making decisions and negotiating.
2. There may need to be differentiation in decision making, which means decentralising the function in customisation projects involving multiple distributed clients to meet the purpose of creating contrasting levels of decisions among different clients (Int. 1)

These findings have confirmed the second hypothesis (Hypothesis 2) and answered Research Question 2.

7.5 Local Development Process in Customisation Projects

The second scenario of CCRD model (Figure 4.2) is presented in section 4.4.2, and involves enhanced onsite development practices, beyond taking decisions at the client's location. It aims to provide a mechanism for managing and communicating customisation requirements for multiple clients across distributed boundaries, and to apply the concept of local development processes that have succeeded in various software development approaches such as Agile (Fowler 2003). Therefore, Section 6.3 presents the evaluation experiments for CCRD mode for local development in customisation projects. This experiment aimed to examine the following hypothesis:

> *Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process.*

The following sections discuss three factors to examine Hypothesis 3, namely decision-making time, development time and entire duration of the customisation process.

7.5.1 Impact of local development on decision time

The results of the experiment, as presented in section 6.3.3, show the statistically significant reduction in the duration of decision making for customisation requirements when decisions are made at a client's location on some requirements, alongside a development team (Table 6.3 and Figure 6.8). This experiment attempted to report only on requirements that did not involve changes to the system functions, as discussed in section 6.3.1. The reasons for the shorter period when decisions are made locally have already been discussed in section 7.4.1. Figure 7.1 reveals a significant reduction in decision-making duration for the three groups of clients involved in both experiments. However, there is also a clear impact on decision making time by enabling local development, as illustrated in Figure 7.1. The reason is that local decision making affects decision-making duration, as confirmed in section 7.4.1 and shown in Table 6.1 and Table 6.3. Undertaking some development processes at a client's location reduces development time, as discussed in section 07.2, but does not change any other aspects of the decision-making process. However, participants in the confirmatory study (section 5.5) indicate benefits from holding discussions between clients, decision makers and developers, aiding an understanding of the

requirements and providing clients with an accurate estimation of anticipated development time, as mentioned by Int. 3 and Int. 5 in section 6.4.4. Int. 4 comments: '*We had a project with a decision maker and a developer in the client's location and the response and the communications was better than now.*' In addition, the literature includes many examples of the success of Agile methods, which rely on local teams including a decision maker, development and customer, such as Scrum (Schwaber, 2004; Cocco et al., 2011).

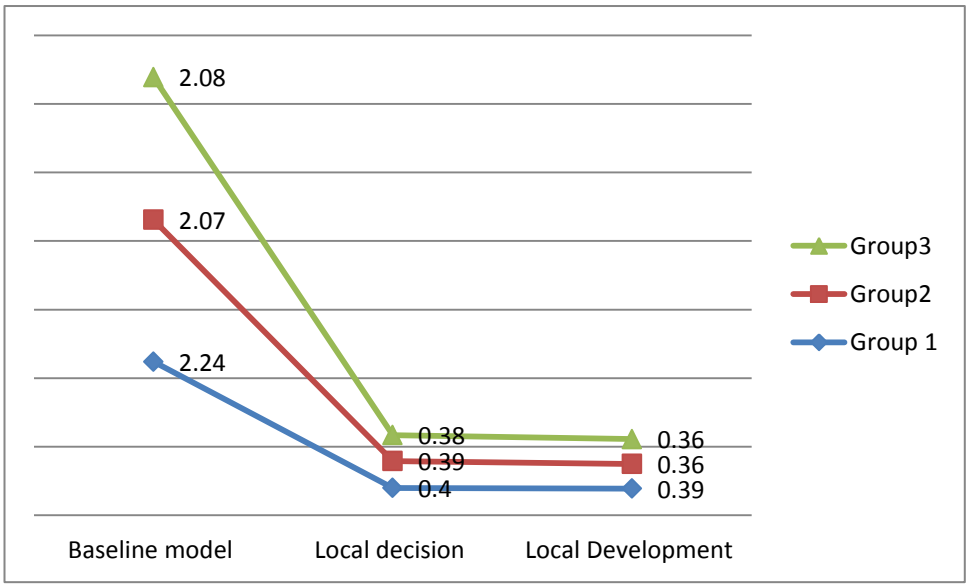


Figure 7.1: Comparison of means of decision-making duration under baseline model and with local decision and local development practices

7.5.2 Impact of local development on development process time

This research aims to discuss the impact of locality on the customisation process in order to reduce the challenges of communicating customisation requirements across the distributed domain. This is presented in the CCRD model for both local decision making (Figure 4.1) and local development (Figure 4.2). Allocating some of the development processes to the client's location, besides decision making, reduces the duration of the development significantly, as shown in Table 6.4 and Figure 6.9. This indicates a reduction in the mean development duration for all three groups involved in this experiment (section 6.3.4). Figure 7.2 shows a clear reduction in development duration for the three groups of clients involved in both experiments, for both local decision and local development. The reason for this reduction is that some requirements are developed and completed locally, reducing the time spent in queueing and central development. Furthermore, the confirmatory study findings (section 6.4) indicate that development practices besides decision making should be allocated to a client's location to achieve faster and better understanding and resolution of a client's requirements through face-

to-face discussion (Int. 4 and Int. 5). Moreover, other participants indicate that some customisation requirements are solved in only a short time yet take a long time queueing for central development, dependent on their priority and the importance of other clients' requirements (Int. 6 and Int. 5).

There were several concerns raised by the interviewees over applying development processes at clients' locations. These are the challenges of using a single-system version for multiple clients, and the motivations of different teams and the difficulty in deciding which requirements to send to the central team and which to apply locally, as indicated by Int. 1. Although four of the seven interviewees had concerns over the effect of local development on single version and its implications, discussion of this issue has just two perspectives on ameliorating its implications.

1. For projects with systems that do not have a single version, a CCRD model with local development will reduce the duration of the development process, as discussed above and as confirmed by some interviewees, such as Int. 4, working on a web system without a single version. He stated: '*In the web application there is no single version. So, the proposed model will be better in time with fewer challenges in communications.*'
2. For projects with a single version, the CCRD model will be useful and reduce the duration of development and the challenges of the distributed domain if reports requirements are allocated locally, do not require functional change and have no need for source code amendment. This suggestion was supported by five of the seven interviewees, such as Int. 7 who states:

   '*Local development is a good idea to reduce the time of development and entire customisation process; we used this with one client, we located a developer to complete his reports requests and that was very useful and faster*'.

In addition, allocating a development team to the client's location for reports requirements would also overcome the second interviewee's concern, namely difficulty in deciding which requirements to send to the central team and which to apply locally, by allocating all such requirements to the local development team.

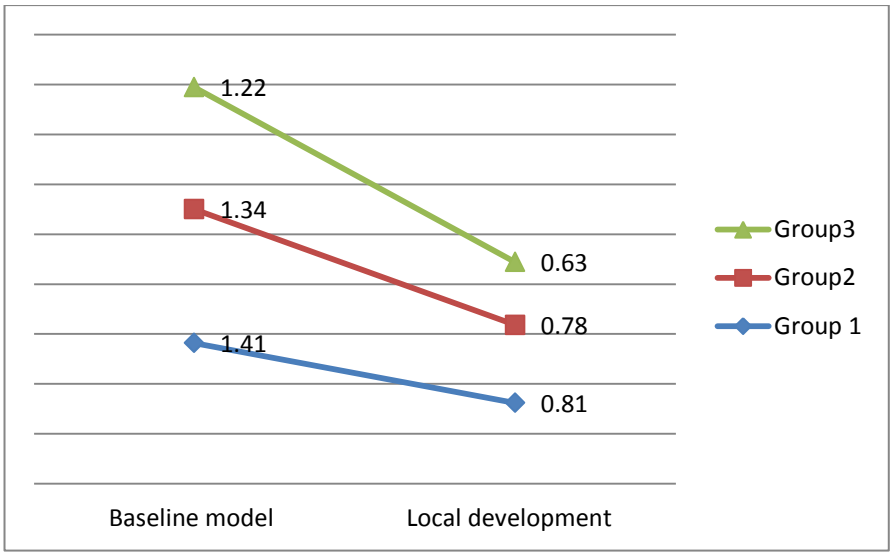


Figure 7.2: Comparison of mean duration of development process under the baseline model and when practising local development

To sum up, allocating development processes for some development customisation requirements, such as reports requirements, has many benefits for the duration of the development process. Moreover, it increases the client's awareness and the developer's understanding of the customisation requirements, and minimises the challenges of communicating requirements across distributed boundaries.

7.5.3 Impact of local development on total duration of customisation process

Undertaking some development at clients' locations reduces the duration of decision making and the development process, as discussed in sections 7.5.1 and 7.5.2 respectively. Therefore, it influences the mean duration of the total customisation process, as shown in Table 6.5. This reveals a significant reduction in the duration of customisation for the three groups involved in the evaluation experiment for the CCRD model, undertaking local development practices at clients' locations. The mean duration of the total customisation process using the CCRD model is half that of the baseline model, under which the entire development process is undertaken off-site by distributed teams (see Figure 7.3).

Furthermore, the confirmatory study's findings (section 6.4.4) indicate improvements to the entire customisation duration by reducing the time taken for customisation through allocating other development practices, beside decision making, to a client's location with a developer on site to deal with some requirements (Int. 5). Participants in the confirmatory study raised the issue of the long wait for simple requirements that take only a few minutes to develop but are delayed in the decision-making or other queues while the central development team serves other clients, as indicated by Int. 6 in section 6.4.4. Thus, allocating development practices locally for

some customisation requirements secures a significant reduction in customisation duration, as shown in Figure 7.3, when both local decision making and local development are employed by the three clients groups involved in the CCRD model evaluation experiments.

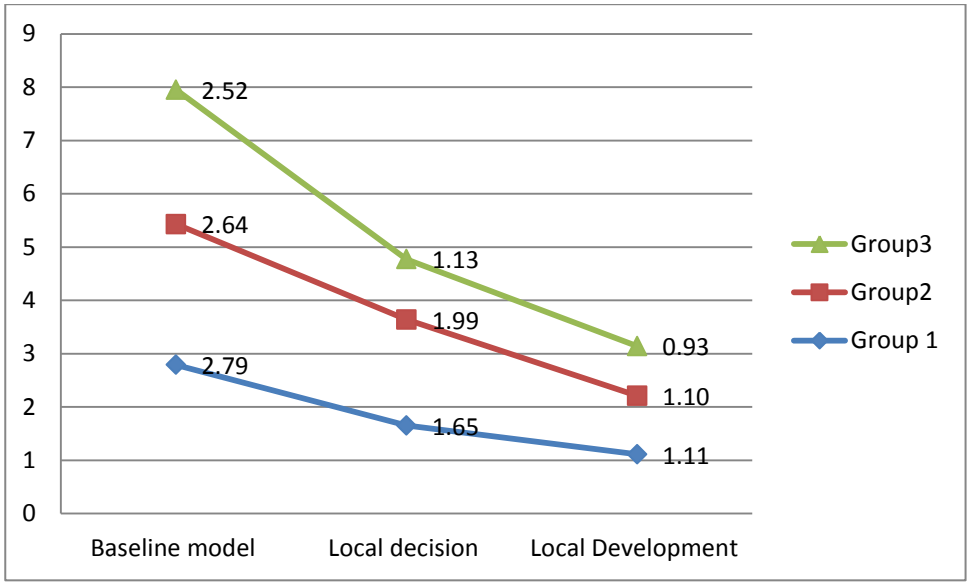


Figure 7.3: Comparison of mean duration of entire customisation under the baseline model, with local decision-making and with local development practices

In summary, this section discusses the impact of local development on the duration of the entire customisation duration. The CCRD model with local development practices (Figure 4.2) has an impact on the duration of decision making, development and the entire customisation. These findings confirm Hypothesis 3 (*Applying development processes for small tasks locally will reduce the development time of the customisation process and the entire customisation process.*), and answers Research Question 3 (*By how much would applying the process of development of minor tasks at clients' locations reduce the total development and customisation time?)*

7.6 Limitations

This section presents the research limitations.

The first limitation is in the preliminary study conducted to confirm the challenges and factors of the customisation process across organisational and distributed boundaries identified from a review of the literature (section 3.3). There were 19 expert participants involved in the study, which may not be considered a large number of subjects for a survey; several issues lay behind this choice of sample size. First, although few, all the participants are highly experienced in the study domain. Second, the main goal of this questionnaire was deductive, to confirm the

findings from literature, and there was no inductive purpose that would demand a larger sample. In addition, based on the G* power run, the sufficient sample size of those parameters is 11. As 19 participants were involved, this increased the power of the study, according to Field (2009).

Another limitation of the evaluation of the CCRD model is its use of simulation. It is difficult to evaluate the CCRD model in a real situation, and the design of experiments tried to reduce the challenge by running a simulation, using real data from a case study through a contextual study (section 5.3) and making comparisons with the baseline of that case study (section 5.4.5). In addition, the findings of simulation experiments were confirmed by seven experts from the same case study in a confirmatory study, using semi-structured interviews (section 5.5 and 6.4).

The third limitation is the use of a single case study in this research. It would be extremely difficult in the time available to this research to find more software development companies that would allow the collection of real data about their clients and the processes of their requirements. Although this research uses a single case study, the selected case study had 18 distributed clients and 2479 requirements in the short observation period (section 5.3.1). However, the case study was used as a baseline for the evaluation experiments (section 5.4.4) and to confirm the findings of those experiments, not in order to build a theory, as in inductive research (section 5.5).

7.7 Chapter Summary

This chapter has discussed the research findings, starting from the identified challenges of the software customisation process for projects undertaken across organisational and distributed boundaries. This was through discussion of FCCSD and the preliminary confirmatory study conducted to confirm the framework components. The results confirmed the challenges of customisation process in the distributed domain using local practices and answering, as shown in section 7.2, Research Question 1: *What are the challenges identified in the literature in customising software products across organisational boundaries in the distributed domain?*

In addition, this has chapter discussed the findings of the evaluation experiments for the CCRD model both for local decision making and local development. These results showed statistically significant reductions in the duration of decision making, development and the entire customisation process under the CCRD model's two scenarios (local decision and local development). Furthermore, these are supported by the findings of the confirmatory study, which emphasised the results shown in sections 7.4 and 7.5, answering Research Questions 2 and 3.

RQ2: What is the impact of making decisions locally on the total decision period and the total customisation period of software customisation processes in the distributed domain? (Answered in section 7.4)

RQ3: How much would undertaking the process of development of minor tasks at clients' locations reduce the total development and customisation period? (Answered in section 7.5).

It also confirms the defined hypotheses.

Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients. (Discussed in section 7.4)

Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients. (Discussed in section 7.4)

Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process. (Discussed in section 7.5)

Finally, this chapter discussed the main research limitations, which were the small number of participants in the preliminary study to confirm the FCCSD, the use of a simulation approach to evaluate the CCRD model and using a single case study. All these limitations have been explained and discussed in section 7.6.

Chapter 8: Conclusion and Future Work

8.1 Introduction

This chapter provides the conclusion to this study. Section 8.2 presents the objectives of the research into the customisation of software products across distributed boundaries, giving an overview of the activities undertaken by summarising the main points of each previous chapter. The findings are presented in section 8.3, answering the research questions and hypotheses. Finally, section 8.4 suggests future work arising from this research.

8.2 Research Summary

The aim of this study has been to identify the challenges and factors from literature for the customisation of software products across organisational and distributed boundaries using Agile software practices, and to provide a framework (FCCSD) with which to address these challenges. In addition, the research discusses the impact of local decision making and development on the customisation process by providing a model for communicating the customisation requirements of multiple clients in the distributed domain (CCRD). These goals are achieved through the stages and activities listed in the following sections.

8.2.1 Build and confirm the FCCSD framework

This research started by exploring the literature on customising software across organisational boundaries to identify challenges and factors presented in Chapter 2. This has been to inform researchers and practitioners of the key issues for customisation projects, especially those adopting local practices, as in the Agile development approach, with multiple clients in the distributed domain. These challenges are presented in the form of a framework termed the FCCSD, in four main categories based on the type of challenge and its impact, namely communication, project management, knowledge management and configuration management, as explained in section 3.2. The design of the FCCSD was followed by a preliminary study conducted to confirm the FCCSD components with 19 experts in the area of software development (section 3.3).

8.2.2 Design and evaluate CCRD model

The CCRD model was designed to model the communication essential to any customisation for distributed clients. It aimed to enhance local decision making and development in order to

overcome the challenges of communicating clients' requirements across multiple sites, basing the customisation process on practices that are typical in the software development life cycle. The model considered two scenarios. The first presented the communication of customisation requirements for multiple clients using local decision making, as shown in Figure 4.1. The second presented both local decisions and local development of certain requirements, as shown in Figure 4.2.

Both scenarios were evaluated using a case study of a company with 18 distributed clients and 2479 requirements over a short observation period (section 5.3.1). The evaluation process went through a sequence of activities, as follows.

8.2.2.1 Contextual review study

The evaluation of the CCRD model began by selecting the case (section 5.3), starting with a contextual review using semi-structured interviews with two experts to collect both information relating to customisation and the challenges of communicating customisation requirements in the distributed domain, and historical data to enable the construction and running of simulation models (section 5.3). The outcome of this study was information on the customisation of multiple distributed clients' requirements in order to build a conceptual model of the selected case study for use as a baseline in evaluation experiments. Moreover, the historical data of requirements and the processes applied were used to analyse it and integrate it into the case study simulation as trace data (section 5.3.5).

8.2.2.2 CCRD model evaluation experiments

The second stage involved simulating the CCRD model using a discrete-event simulation approach, traced in the baseline model using real data collected during the contextual review (section 5.4). This stage included two experiments, one to evaluate the CCRD model using local decision making (section 6.2) and the second to evaluate it using both local decision making and local development practices (section 6.3). Both were conducted using real data from 18 clients, divided into three groups based on their numbers of requirements over the observation period. Figure 6.3 illustrates these client groups and the sequence of the experiments for each. The outcomes of this stage were statically significant results for the CCRD model using local decision making and local development to reduce the mean duration of the decision making, the development and the entire customisation processes compared with the baseline model presented in Chapter 6 and discussed in Chapter 7.

8.2.2.3 Confirmatory study

The findings of the two evaluated experiments for the CCRD model (in the previous section) were confirmed by seven experts at the case study through a confirmatory study (section 5.5), using semi-structured interviews to obtain their opinions on using local decision making and local development practices in customising software products for multiple clients across distributed boundaries. This study supported the results of experiments to answer the research questions and concurred with the defined hypotheses (section 6.4). The summary outcomes were strong agreement on experimental results and the use of local decision making and local development. Moreover, participants made some suggestions, as discussed in sections 7.4 and 7.5.

8.3 Research Findings

This section presents a summary of the research findings, based on the following research questions and defined hypotheses:

> *RQ1: What are the challenges identified in the literature in customising software products across organisational boundaries in the distributed domain?*

This research question aimed to investigate the challenges identified in the literature on customising software products across organisational boundaries in the distributed domain. This question has been answered by a confirmed framework termed FCCSD that synthesises the findings in the literature on RE, DSD and the Agile development approach to the customisation process, as presented in Chapter 3 and discussed in Chapter 7. The FCCSD addresses the key challenges and factors of customising products across organisational boundaries in the distributed domain to help researchers and practitioners understand and consider these challenges.

> *RQ2: What is the impact of making decisions locally on the total decision period and the total customisation period of software customisation processes in the distributed domain?*

This research question aimed to investigate the impact of making decisions at a client's location on the duration of both decision making and the entire customisation process for the distributed multiple-client domain. This question has been answered by confirming the two defined hypotheses:

Hypothesis 1: Allocating decision making to clients' locations will reduce the total decision-making duration for customising the requirements of distributed clients.

The results of the evaluation of the CCRD model on local decision making for clients' requirements to reduce the decision making period in distributed multi-clients projects were statistically significant and supported by the findings of the confirmatory study of this hypothesis, as presented in Chapter 6, section 6.2 and discussed in Chapter 7, section 7.4.1.

Hypothesis 2: Allocating decision making to clients' locations will reduce the impact of communication challenges by reducing the total duration of the entire customisation process to satisfy the requirements of distributed clients.

Changing any activity in the process of customisation, from collecting requirements through to delivering solutions, affects the length of the entire process. This hypothesis has been confirmed by a statistically significant result that showed the beneficial impact of local decision making on the duration of the entire customisation process for multiple-client projects in the distributed domain, supported by the results of the confirmatory study presented in Chapter 6, section 6.2 and discussed in Chapter 7, section 7.4.2.

RQ3: How much would undertaking the process of development of minor tasks at clients' locations reduce the total development and customisation time?

This research question aimed to investigate the impact of allocating minor customisation requirement development to the client's location on the length of the development period and the entire duration of customisation in the distributed multiple-client domain. This research has been answered by confirming the hypothesis defined for its solution:

Hypothesis 3: Applying development processes for some tasks locally would decrease the time for development in customisation and thus the duration of entire customisation process.

The statistically significant results of evaluating the CCRD model for local development process, presented in Chapter 6, section 6.3 and discussed in Chapter 7, section 7.5, indicated that its impact is to reduce the duration of both the development and the entire customisation, supported by the findings of the confirmatory study, which confirmed this hypothesis and answers Research Question 3.

8.4 Future Work

This section suggests a number of ways to extend this research in terms of new research ideas or improvements to the simulation models of this research for other purposes.

8.4.1 Metrics of the level of challenge of customisation projects in the DSD domain

The FCCSD addresses the challenges faced by researchers and practitioners in customisation projects involving multiple clients across organisational and distributed. The framework can be extended by developing metrics to gauge these challenges in this domain, assisting project managers to anticipate the level of difficulty and to design plans to overcome the issues.

8.4.2 Evaluation of the impact of local decision making and development practices

The evaluation of decision making and development practice at a client's location can be extended in many ways. This section provides some suggestions.

8.4.2.1 Evaluate the locality of various sizes of companies to examine the impact

This research has used a single case study with 18 clients, involving 2479 requests concerning the distributed clients' requirements covering 1290 working hours. Future work might examine local decision making and development practices at clients' locations using other case studies to consider various sizes of project and different numbers of customisation requirements in order to compare the impact.

8.4.2.2 Consider different types of customisation requirements in the local development process

The customisation requirements examined in this study to evaluate the development practices at the clients' location were the building or fixing of reports, since this neither involves change to the main system code nor affects clients' unique system versions. Therefore, one possibility is to find a way to assign local development of various types on the basis of having a single-system version. This suggestion can be progressed by establishing a central repository for source code and designing a model to enhance the updating process for distributed teams.

A further suggestion to widen the use of the simulations is to construct the baseline model using real data to reflect the process of customisation for multiple distributed clients (section 5.4). It can be extended to include cost–benefit analysis to measure the efficiency of simulate projects of customisation process. This process needs real data or empirical estimations of resource costs.

8.5 Conclusion

The research has investigated the customisation of software products for multiple clients across distributed boundaries. Challenges of that domain were identified from the literature and addressed by a confirmed framework termed the FCCSD. This provided the key challenges and factors of the customisation process for multiple-client projects across organisational and distributed boundaries when local development practices such as the Agile software approach are used, to be considered by researchers and practitioners of software customisation projects.

Furthermore, this research has focused on communicating customisation requirements in order to reduce the challenges of describing these requirements across the distributed domain, proposing a model termed CCRD. The findings of the model's evaluation experiments, both for local decision making and local development, show a statistically significant reduction in the duration of decision making, development and the entire customisation process under its two scenarios (local decision and local development). The results of the evaluation experiments were supported by the findings of the confirmatory study, indicating a statistically significant impact on decision making and negotiation of clients' requirements. This goes beyond conducting customisation development at clients' premises and reducing delays in coming to decisions, in the development process and in the entire customisation; it minimises holdups due to misunderstandings over clients' requirements across distributed boundaries.

List of References

Abdel-Hamid, T. K., 1988. The economics of software quality assurance: A simulation-based case study. *MIS Quarterly*, 12, 395–411. Available at: http://www.jstor.org/stable/249206\nhttp://www.jstor.org/stable/pdfplus/249206.pdf.

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J., 2002. *Agile Software Development Methods: Review and analysis.* VTT Technical Report 478, online at http://www2.vtt.fi/inf/pdf/publications/2002/P478.pdf

Achimugu, P., Selamat, A., Ibrahim, R., & Mahrin, M. N., 2014. A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56, 568–585.

Ahmad, A., Shahzad, A., Padmanabhuni, V. K., Mansoor, A., Joseph, S., & Arshad, Z., 2011. Requirements prioritization with respect to geographically distributed stakeholders. *2011 IEEE International Conference on Computer Science and Automation Engineering*, pp. 290–294.

Akbar, R., Harris, M., & Naeem, M., 2008. Agile framework for globally distributed development environment (The DAD Model). In *8th WSEAS International Conference on Applied Informatics and Communications*. Rhodes, Greece, pp. 423–428.

Alfonso, A., Braberman, V., & Kicillof, N., 2004. Visual timed event scenarios. In *Proceedings. 26th International Conference on Software Engineerin*g (ICSE 2004). IEEE Computer Society Press.

Ali Babar, M., Verner, J. M., & Nguyen, P. T., 2007. Establishing and maintaining trust in software outsourcing relationships: An empirical investigation. *Journal of Systems and Software*, 80, 1438–1449.

Alur, R., Etessami, K., & Yannakakis, M., 2003. Inference of message sequence charts. *IEEE Transactions on Software Engineerin*g, 29(7), 623–633.

Andrea, J., 2007. Envisioning next-generation functional testing tools. *IEEE Software*, 24, 58–66.

Argyrous, G., 2011. *Statistics for Research: With a guide to SPSS*, 3rd edn. London: Sage.

Atkinson, R., 1999. Project management: Cost, time and quality, two best guesses and a phenomenon, it's time to accept other success criteria. *International Journal of Project Management*, 17(6), 337–342.

Azam, F., Qadri, S., Ahmad, S., Khan, K., Siddique, A. B., & Ehsan, B., 2014. Framework of software cost estimation by using object orientated design approach., 3(8), 97–100.

Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M., 2004. *Discrete-Event System Simulation*. New Jersey: Prentice Hall.

References

Berander, P. & Andrews, A., 2005. Requirements prioritization. In P. Berander & A. Andrews (eds), *Engineering and Managing Software Requirements*, pp. 69–94. Berlin: Springer.

Beyer, H. & Holtzblatt, K., 1998. *Contextual Design: Defining customer-centered systems*. San Francisco: Morgan Kaufmann.

Bhat, J. M., Gupta, M., & Murthy, S. N., 2006. Overcoming requirements engineering challenges: Lessons from offshore outsourcing. *IEEE Software*, 23, 38–44.

Bjarnason, E., Wnuk, K., & Regnell, B., 2011. A case study on benefits and side-effects of Agile practices in large-scale requirements engineering. *Proceedings of 1st Workshop on Agile Requirements Engineering - AREW '11*, pp. 1–5.

Bosch, J. & Bosch-Sijtsema, P., 2010. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1), 67–76. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0164121209001617 [Accessed October 22, 2014].

Bowen, S. & Maurer, F., 2002. Process support and knowledge management for virtual teams doing Agile software development. *Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC 2002)*, pp. 1–3.

Boyatzis, R.E., 1998. *Transforming Qualitative Information: Thematic Analysis and Code Development*. Thousand Oaks, CA: Sage.

Brinkkemper, S., Ebert, C., & Versendaal, J., 2006. *Proceedings of the First International Workshop on Software Product Management.* 2006 International Workshop on Software Product Management (IWSPM'06 - RE'06 Workshop), September, Minneapolis, pp. 1–2.

Brownsword, L., Oberndorf, T., & Sledge, C.A., 2000. Developing new processes for COTS-based systems. *IEEE Software*, 17, 48–55.

Bush, A., Tiwana, A., & Tsuji, H., 2008. An empirical investigation of the drivers of software outsourcing decisions in Japanese organizations. *Information and Software Technology*, 50(6), 499–510. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0950584907000869 [Accessed November 3, 2014].

Calefato, F., Damian, D., & Lanubile, F., 2007. An empirical investigation on text-based communication in distributed requirements workshops. In *Proceedings - International Conference on Global Software Engineering, ICGSE 2007*, pp. 3–11.

Campbell, L.A., Cheng, B. H. C, McUmber, W. E., & Stirewalt, K., 2002. Automatically detecting and visualising errors in UML diagrams. *Requirements Engineering*, 7, 264–287.

Carmel, E. & Agarwal, R., 2006. The maturation of offshore sourcing of information technology work. In R. Hirschheim, A. Heinzl & J. Dibbern (eds), *Information Systems*

Outsourcing: *Enduring themes, new perspectives and global challenges*, 2nd edn, pp. 631–650. Berlin: Springer.

Carmel, E. & Tjia, P., 2005. *Offshoring information technology: Sourcing and outsourcing to a global workforce*. Cambridge: Cambridge University Press.

Carrillo de Gea, J. M., Nicolás, J., Fernández Alemán, J. L., Toval, A., Ebert, A., & Vizcaíno, A., 2012. Requirements engineering tools: Capabilities, survey and assessment. *Information and Software Technology*, 54(10), 1142–1157. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0950584912000766 [Accessed November 6, 2014].

Cheng, B. H. C. & Atlee, J. M., 2009. Research directions in requirements engineering. *Lecture Notes in Business Information Processing*, 14, 11-43.

Chow, A. & Joy, M., 2005. Shifting the focus from methodologies to techniques. In *6th Annual Conference of the HEA Network for Information and Computer Sciences*. York, UK, pp. 25–29.

Chwif, L., Barretto, M. R. P., & Paul, R. J., 2000. On simulation model complexity. In *Proceedings of 2000 Winter Simulation Conference*, Orlando, DOI 10.1109/WSC.2000.899751, vol. 1, pp. 449–455.

Cocco, L., Mannaro, K., Concas, G., & Marchesi, M., 2011. Simulating Kanban and Scrum vs. Waterfall with system dynamics. In Proceeding of 12th International Conference, XP 2011, May, Madrid, *Agile Processes in Software Engineering and Extreme Programming*. Berlin: Springer, pp. 117–131.

Cohen, D., Lindvall, M. & Costa, P., 2004. An introduction to Agile methods. *Advances in Computers*, 62(03), 1–66.

Cohen, L. & Manion, L., 1997. *Research Methods in Education*, 8th edn. London: Routledge.

Cohene, T. & Easterbrook, S., 2005. Contextual risk analysis for interview design. In *Proceedings of 13th IEEE International Conference on Requirements Engineering (RE'05)*, August-September, Paris.

Colomo-Palacios, R., Casado-Lumberos, C., Soto-Acosta, P., García-Peñalvo, F. J., & Tovar, E., 2014. Project managers in global software development teams: A study of the effects on productivity and performance. *Software Quality Journal*, 22(1), 3–19. Available at: http://link.springer.com/10.1007/s11219-012-9191-x [Accessed October 28, 2014].

Colomo-Palacios, R., Soto-Acosta, P., & García-Peñalvo, F. J., 2012. A study of the impact of global software development in packaged software release planning. Journal of Universal Computer Science,18(19), 2646–2668.

Concannon, K. H., Hunter, K. I. & Tremble, J. M., 2003. SIMUL8-Planner simulation-based planning and scheduling. *Proceedings of the 2003 International Conference on Machine*

Learning and Cybernetics (IEEE Cat. No.03EX693), pp. 1488–1493. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1261593.

Coram, M. & Bohner, S., 2005. The impact of Agile methods on software project management. In *Proceedings of 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05),* April, Greenbelt, Maryland., pp. 363–370.

Creswell, J. W., 2008. *Research Design: Qualitative, quantitative, and mixed methods approaches.* p. 398. Thousand Oaks, CA: Sage.

Dahlstedt, Å. G., Karlsson, L., Persson, A., Natt och Dag, J., & Regnell, B., 2003. Market-driven requirements engineering processes for software products – a report on current practices. In *International Workshop on COTS and Product Software RECOTS, in conjunction with 11th IEEE International Requirements Engineering Conference*. September, Monterey,

Damas, C., Lambeau, B. & van Lamsweerde, A., 2006. Scenarios, goals, and state machines: A win-win partnership for model synthesis. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering,* pp. 197–207.

Damian, D., 2002. The study of requirements engineering in global software development: As challenging as important. In *Proceedings of International Workshop on Global Software Development,* Florida.

Damian, D., 2006. Requirements engineering in distributed projects. In *Proceedings of 2006 IEEE International Conference on Global Software Engineering (ICGSE'06)*, pp. 69–69.

Damian, D., 2007. Stakeholders in global requirements engineering: Lessons learned from Practice. *IEEE Software*, March/April, 21–27.

Damian, D. & Moitra, D., 2006. Guest editors' Introduction: Global software development: how far have we come? *IEEE Software*, 23.

Damian, D. & Zowghi, D., 2002a. An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. In *Proceedings of 36th Hawaii Conference on Systems Sciences (HICSS'36),* Hawaii, January.

Damian, D. & Zowghi, D., 2002b. The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In Proceedings of IEEE Joint International Conference on Requirements Engineering, pp. 319–328.

Damian, D. & Zowghi, D., 2003. Requirements engineering challenges in multi-site software development organizations. *Requirements Engineering Journal*, 8, 149–160.

Damian, D., Zowghi, D., Vaidyanathasamy, L., & Pal, Y., 2004. An industrial case study of immediate benefits of requirements engineering process improvement at the Australian Center for Unisys Software. *Empirical Software Engineering*, 9, 45–75.

Damian, D., Izquierdo, L., Singer, J., & Kwan, I., 2007. Awareness in the wild: Why communication breakdowns occur. In *Proceedings of the International Conference on Global Software Engineering, ICGSE 2007*, pp. 81–90.

da Silva, F. Q. B., França A. C. C., & Prikladnicki, R., 2010. Challenges and solutions in distributed software development project management: A systematic literature review. In *5th IEEE International Conference on Global Software Engineering*. IEEE, pp. 87–96.

Davis, P. K., 1992. *Generalizing Concepts and Methods of Verification, Validation, and Accreditation (VV&A) for Military Simulations*. Santa Monica, CA: Rand Corp.

del Nuevo, E., Piattini, M. & Pino, F. J., 2011. Scrum-based methodology for distributed software development. *2011 IEEE 6th International Conference on Global Software Engineering*, pp. 66–74. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6063151 [Accessed February 12, 2013].

Denzin, N. & Lincoln, Y., 2005. *The Sage Handbook of Qualitative Research.* Thousand Oaks, CA: Sage.

Dingsøyr, T., Dybå, T., & Abrahamsson, P., 2008. A preliminary roadmap for empirical research on Agile software development. In *Proceedings of Agile 2008 Conference*. IEEE, pp. 83–94.

Dybå, T. & Dingsøyr, T., 2009. What do we know about Agile software development? *IEEE Software*, 26(5), 6–9.

Ebert, C., 2007. Optimizing supplier management in global software engineering. In *International Conference on Global Software Engineering ICGSE 2007*, pp. 177–185.

Ebert, C., 2012. *Global software and IT: A guide to distributed development, projects, and outsourcing*. Los Angeles: Wiley-IEEE Computer Society Press.

Ebert, C., Dumke, R., Bundschuh, M., & Schmietendorf, A., 2005. *Best Practices in Software Measurement: How to use metrics to improve project and process performance*, chap. 4. Heidelberg: Springler-Verlag.

Erickson, J., Lyytinen, K., & Siau, K., 2005. Agile modeling, Agile software development, and extreme programming: The state of research. *Journal of Database Management*, 16(4), 88–90.

Espinosa, J., Slaughter, S. A., Kraut, R. E., & Herbsleb, J. D., 2007. Team knowledge and coordination in geographically distributed software development. *Journal of Management Information Systems*, 24, 135–169.

Evans, D., Gruba, P., & Zobel, J., 2011. *How To Write a Better Thesis*. Carlton, Australia: Melbourne University Press.

References

Faiz, M. F., Qadri, U. & Ayyubi, S. R., 2007. Offshore software development models. *2007 International Conference on Information and Emerging Technologies*, pp. 1–6. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4381329.

Fente, J., Knutson, K., & Schexnayder, C., 1999. Defining a Beta distribution function for construction simulation. *WSC'99. 1999 Winter Simulation Conference Proceedings. Simulation - A Bridge to the Future* (Cat. No.99CH37038).

Fenton, N. E. & Pfleeger, S.L., 1997. *Software Metrics: A rigorous and practical approach*. Boston: PWS.

Field, A., 2009. *Discovering Statistics Using SPSS*, 3rd edn. London: Sage.

Fink, A., 2003. *How to Ask Survey Questions*, 2nd edn. London: Sage.

Fogelström, N. D., 2010. *Understanding and Supporting Requirements Engineering Decisions in Market-driven Software Product Development*. School of Computing, Blekinge Institute of Technology.

Fowler, M., 2001. The new methodology. *Wuhan University Journal of Natural Sciences*, 6(1-2), 12–24.

Fowler, M., 2003. Using an Agile software process with offshore development. Online at http://martinfowler.com/articles/agileOffshore.html

Fowler, M. & Highsmith, J., 2001. *The Agile Manifesto.* San Francisco, CA: Miller Freeman.

Gaba, D. M., 2004. The future vision of simulation in health care. *Quality & Safety in Healthcare*, 13 Suppl 1, i2–i10.

Glass, R. L., 1982. *Modern Programming Practices: A report from industry*. Englewood Cliffs, NJ: Prentice Hall.

Glass, R. L., Vessey, I. & Ramesh, V., 2002. Research in software engineering: An analysis of the literature. Infor*mation and Software Technology*, 44(8), 491-506.

Gopal, A., Alberto, E. J., Sanjay, G., & David, D., 2011. Coordination and performance in global software service delivery: The vendor's perspective. *IEEE Transactions on Engineering Management*, 58(4), 772–785.

Gopalakrishnan S., K. V. P. Y. S., 1996. Offshore model for software development: The infosys experience. In *Proceedings of the ACM SIGCPR Conference*, pp. 392–393.

Gray, D. E., 2004. *Doing Research in the Real World*. London: Sage.

Gray, D. E., 2009. *Doing Research in the Real World*, 2nd edn. London: Sage.

Gregory, T. & Baskerville, R., 2014. Traveling of requirements in the development of packaged software: An investigation of work design and uncertainty. Dissertation, Georgia State University, http://scholarworks.gsu.edu/cis_diss/53

Gupta, A., 2009. Deriving mutual benefits from offshore outsourcing. *Communications of the ACM*, 52(6), 122-126.)

Hayat, F. N., Ehsan, N., Ishaque, S. Ahmed, S., & Mirza, E., 2010. A methodology to manage the changing requirements of a software project. *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, pp. 319–322.

Hayata, T. & Han, J., 2011. A hybrid model for IT project with Scrum. In *Proceedings of Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference*., pp. 285–290.

Herbsleb, J. D., Grinter, R. E., & Finholt, T. A., 2001. An empirical study of global software development: Distance and speed. In *ICSE, IEEE*, pp. 81–90. Toronto.

Herbsleb, J. D. & Mockus, A., 2003. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, 29(6), 481–494. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1205177.

Herbsleb, J. D. & Moitra, D., 2001. Global software development - IEEE Software. *Focus, IEEE Software*, April, 16–20.

Hernández-López, A., Colomo-Palacios, R., García-Crespo, Á., & Soto-Acosta, P., 2010. Team software process in GSD teams: A study of new work practices and models. *International Journal of Human Capital and Information Technology Professionals*, 1(3), 32–53.

Höfner, G. & Mani, V. S., 2007. TAPER: A generic framework for establishing an offshore development center. In *Proceedings - International Conference on Global Software Engineering, ICGSE 2007*, pp. 162–172.

Holmström, H., Conchúir, E. Ó., Agerfalk, P. J., Fitzgerald, B., 2006. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In Proceedings of *International Conference on Global Software Engineering, 2006. ICGSE '06*, pp. 3–11.

Holmström, H., Fitzgerald, B., Ågerfalk, P. J., & Conchúir, E. Ó., 2006. Agile practices reduce distance in global software development. *Information Systems Management*, 23(3), 7–18. Available at: http://www.tandfonline.com/doi/abs/10.1201/1078.10580530/46108.23.3.20060601/93703.2 [Accessed November 5, 2014].

Holtzblatt, K., Wendell, J. & Wood, S., 2005. *Rapid Contextual Design: A how-to guide to key techniques for user-centered design*. San Francisco: Morgan Kaufmann.

Hossain, E., Babar, M. A., & Paik, H., 2009. Using Scrum in global software development: A systematic literature review. *4th IEEE International Conference on Global Software Engineering*, pp. 175–184.

References

Hossain, E., Babar, M. A., Paik, H., & Verner, J., 2009. Risk identification and mitigation processes for using Scrum in global software development: A conceptual framework. *16th Asia-Pacific Software Engineering Conference*, pp. 457–464.

Hove, S. E. & Anda, B., 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. *11th IEEE International Software Metrics Symposium (METRICS'05)*, pp. 23–23. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1509301.

Iqbal, J., Ahmed, R. & Marczak, S., 2014. A framework to resolve requirements engineering issues in software development outsourcing. In *Proceedings of IEEE 4th International Workshop on Empirical Requirements Engineering (EmpiRE),* August, Karlskrone, pp. 72–75.

Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L. K., & Young, T., 2010. Simulation in manufacturing and business: A review. *European Journal of Operational Research*, 203, 1–13.

Jalali, S. & Wohlin, C., 2010. Agile practices in global software engineering - A systematic map. *5th IEEE International Conference on Global Software Engineering*, pp. 45–54.

Jebreen, I., 2014. Packaged software implementation requirements engineering by small software enterprises: An ethnographic study. Thesis, Auckland University of Technology.

Jebreen, I. & Wellington, R., 2013. Understanding requirements engineering practices for packaged software implementation. *IEEE 4th International Conference on Software Engineering and Service Science*, pp. 229–234. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6615294.

Jemmott, H., 2008. Using NVivo for qualitative data analysis. *Analysis*, 1, pp. 7–7.

Jiang, J. & Klein, G., 2000. Software development risks to project effectiveness. *Journal of Systems and Software*, 52, 3–10.

Jiao, J. R. & Chen, C., 2006. Customer requirement management in product development : A review of research issues. *Concurrent Engineering: Research and Applications*, 14(3), 1–25.

Jiménez, M., Piattini, M. & Vizcaíno, A., 2009. Challenges and improvements in distributed software development: A systematic review. *Advances in Software Engineering*, Article ID 710971, pp. 1–14. doi: 10.1155/2009/710971.

Joy, M., 2005. Group projects and the computer science curriculum. *Innovations in Education and Teaching International*, 42(1), pp.15–25.

Karlsson, L., Dahlstedt, A. 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, 588–604.

Kellner, M. I., Madachy, R. J., & Ra, D. M., 1999. Software process simulation modeling : Why ? What ? How ? *Journal of Systems and Software*, 46, 91–105.

Kent Beck et al., 2001. Manifesto for Agile software development. Available at: http://agilemanifesto.org/ [Accessed February 14, 2012].

Khan, A. A., Basri, S., & Dominic, P. D. D., 2012. A proposed framework for requirement change management in global software development. *2012 International Conference on Computer & Information Science (ICCIS)*, pp. 944–947.

Khan, S. U., Niazi, M., & Ahmad, R., 2009. Critical success factors for offshore software development outsourcing vendors: A systematic literature review. *4th IEEE International Conference on Global Software Engineering*, pp. 207–216. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5196934 [Accessed November 3, 2014].

Khan, S. U., Niazi, M., & Ahmad, R., 2011. Factors influencing clients in the selection of offshore software outsourcing vendors: An exploratory study using a systematic literature review. *Journal of Systems and Software*, 84(4), 686–699. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0164121210003298 [Accessed August 29, 2014].

Kiani, Z. U. R., Šmite, D., & Riaz, A., 2013. Measuring awareness in cross-team collaborations – distance matters. *IEEE 8th International Conference on Global Software Engineering*, pp. 71–79. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6613070 [Accessed August 12, 2014].

Kircher, M., Jain, P., Corsari, A., & Levine, D., 2001. Distributed extreme programming. In *Proceedings of the International Conference on eXtreme Programming and Flexible Processes in Software Engineering*. pp. 66–71. Sardinia, May.

Kleijnen, J.P.C., 1995. Verification and validation of simulation models. *European Journal of Operational Research*, 82(1), 145-162. Online at https://pure.uvt.nl/portal/files/947119/verifica.pdf

Korkala, M. & Abrahamsson, P., 2007. Communication in distributed Agile development : A case study. In M. Korkala and P. Abrahamsson (eds), *33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 203–210.

Kotonya, G. & Sommerville, I., 1996. Requirements engineering with viewpoints. *Software Engineering Journal*, 11, 5.

Kranzler, J. H., 2003. *Statistics for the Terrified*, 3rd edn. New Jersey: Pearson Education.

Krishna, S., Sahay, S. & Walsham, G., 2006. Managing cross-cultural issues in global software outsourcing. In R. Hirschheim, A. Heinzl & J. Dibbern (eds), *Information Systems*

Outsourcing: *Enduring themes, new perspectives and global challenges*, 2nd edn, pp. 651–658. Berlin: Springer.

Lanubile, F., Mallardo, T., & Calefato, F., 2003. Tool support for geographically dispersed inspection teams. *Software Process Improvement and Practice*, 8, 217–231.

Laplante, P. A., Costello, T., Singh, P., Bindiganavile, S., & Landon, M., 2005. The who, what, why, where, and when of IT outsourcing. *IT Professional*, 6(1), 19-23

Law, A. M., 1991. Simulation-models level of detail determines effectiveness. *Industrial Engineering*, 23(10), 16–18.

Law, A. M., 2007. *Simulation Modeling and Analysis*, 4th edn. New York: McGraw-Hill.

Layman, L., Williams, L., Damian, D., & Bures, H., 2006. Essential communication practices for extreme programming in a global software development team. *Information and Software Technology*, 48, 781–794.

Lee, S. & Yong, H.-S., 2009. Distributed Agile: Project management in a global environment. *Empirical Software Engineering*, 15(2), 204–217.

Lehtola, L., Kauppinen, M., & Kujala, S., 2004. Requirements prioritization challenges in practice. In *Product Focused Software Process Improvement: Proceedings of 5th International Conference, PROFES 2004*, Kansai Science City, Japan, April, pp. 497–508.

Light, B., 2005. Going beyond 'misfit' as a reason for ERP package customisation. *Computers in Industry*, 56, 606–619.

Lunesu, M. I., 2013. Process software simulation model of Lean-Kanban approach. Thesis, University of Cagliari.

Marshall, C. & Rossman, G. B., 1999. *Designing Qualitative Research*, 3rd edn. Thousand Oaks, CA: Sage.

Martignoni, R., 2009. Global sourcing of software development - A review of tools and services. In *Proceedings - 4th IEEE International Conference on Global Software Engineering, ICGSE*, pp. 303–308.

Martin, R. & Ra, D., 2001. Application of a hybrid process simulation model to a software development project, *Journal of Systems and Software*, 59, 237–246.

McLaughlin, L., 2003. An eye on India: Outsourcing debate continues. *IEEE Software*, 20(3), 114–117.

Miles, M. B. & Huberman, A. M., 1994. *Qualitative Data Analysis: An expanded sourcebook.* London: Sage.

Mishra, D. & Mishra, A., 2009. Market-driven software project through agility: Requirements engineering perspective. In *Lecture Notes in Business Information Processing*, vol. 37, pp. 103–112. Berlin: Springer.

Moe, N. B. & Aurum, A., 2008. Understanding decision-making in Agile software development: A case-study. *34th Euromicro Conference Software Engineering and Advanced Applications*, pp. 216–223.

Morabito, V., Pace, S., & Previtali, P., 2005. ERP marketing and Italian SMEs. *European Management Journal*, 23, 590–598.

Morien, R. & Wongthongtham, P., 2008. Supporting Agility in software development projects - Defining a project ontology. *2nd IEEE International Conference on Digital Ecosystems and Technologies*, pp. 229–234.

Mudumba, V. & Lee, O.-K. (Daniel), 2010. A new perspective on GDSD risk management: Agile risk management. *5th IEEE International Conference on Global Software Engineering*, pp. 219–227.

Munson, J. C., 2003. *Software Engineering Measurement*, New York, USA: Auerbach.

Niazi, M., El-Attar, M., Usma, M., & Ikram, N., 2012. GlobReq: A framework for improving requirements engineering in global software development projects: Preliminary results. In *16th International Conference on Evaluation Assessment in Software Engineering (EASE 2012)*, pp. 166–170.

Nisar, M. F. & Hameed, T., 2004. Agile methods handling offshore software development issues. *Proceedings of 8th International Multitopic Conference INMIC 2004*, pp. 417–422.

Noll, J., Beecham, S. & Richardson, I., 2010. Global Software development and collaboration: Barriers and solutions. *ACM Inroads*, 1, 66–78.

Oden, J. et al., 2006. Simulation-based engineering science: Revolutionizing engineering science through simulation. Report of the National Science Foundation Blue Ribbon Panel on Simulation-based Engineering Science, available at http://www.nsf.gov.publications.pub_summ.jsp?eds_key=sbes0506

Oza, N., 2006. An empirical evaluation of client–vendor relationships in Indian software outsourcing companies. Thesis, University of Hertfordshire.

Oza, N. V., Hall, T., Rainer, A., Grey, S. G., 2006. Trust in software outsourcing relationships: An empirical investigation of Indian software companies. *Information and Software Technology*, 48, 345–354.

Paetsch, F., Eberlein, A., & Maurer, F., 2003. Requirements engineering and Agile software development. *WET ICE 2003. Proceedings. 12th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 308–313. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231428.

Patton, M. Q., 1987. *How to Use Qualitative Methods in Evaluation*, SEGA publications Ltd.

Phalnikar, R., Deshpande, V. S., & Joshi, S. D., 2009. Applying Agile principles for distributed software development. In *Proceedings of 2009 International Conference on Advanced Computer Control*, pp. 535–539.

References

Pitangueira, A. M., Maciel, R. S. P., & Barros, M., 2014. Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature. *Journal of Systems and Software*. doi:10.1016/j.jss.2014.09.038. Available online at: http://linkinghub.elsevier.com/retrieve/pii/S0164121214002118 [Accessed November 11, 2014].

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. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0950584912000493 [Accessed October 10, 2014].

Prajapati, B., Dunne, M., & Armstrong, R., 2010. Sample size estimation and statistical power analyses. *Optometry Today* 16. Online at http://www.optometry.co.uk/uploads/articles/statistical%20article.pdf

Pressman, R. S., 2010. *Software Engineering: A practitioner's approach,* 7th edn. London: McGraw-Hill.

Prikladnicki, R. et al., 2007. Distributed software development: Practices and challenges in different business strategies of offshoring and onshoring. *In Proceedings of International Conference on Global Software Engineering (ICGSE)*, pp. 262–274.

Prikladnicki, R. & Audy, J. L. N., 2010. Process models in the practice of distributed software development: A systematic review of the literature. *Information and Software Technology*, 52(8), 779–791. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0950584910000492 [Accessed July 17, 2011].

Prikladnicki, R., Audy, J. L. N. & Evaristo, R., 2004. An empirical study on global software development: Offshore insourcing of IT projects. In *Proceedings of 26th International Conference on Software Engineering*, Edinburgh, pp. 53–58.

Prikladnicki, R., Nicolas Audy, J. L., & Evaristo, R., 2003. Global software development in practice: Lessons learned. *Software Process: Improvement and Practice*, 8(4), 267–281. Available at: http://doi.wiley.com/10.1002/spip.188 [Accessed October 24, 2014].

Psaroudakis, J. E. & Eberhardt, A., 2011. A discrete event simulation model to evaluate changes to a software project delivery process. *IEEE 13th Conference on Commerce and Enterprise Computing*, 113–120. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6046962 [Accessed January 17, 2015].

Qahtani, A. M., Wills, G. B. & Gravell, A. M., 2012. Toward a framework for the development and specialisation of product software across organisational boundaries. In *Proceedings of International Conference on Information Society (i-Society 2012)*. London, pp. 391–395.

Racheva, Z., Daneva, M., Herrmann, A., & Wieringa, R. J., 2010. A conceptual model and process for client-driven Agile requirements prioritization. *4th International Conference on Research Challenges in Information Science (RCIS)*, pp. 287–298.

Ranganathan, C. & Balaji, S., 2007. Critical capabilities for offshore outsourcing of information systems. *MIS Quarterly Executive*, 6, 147–164.

Ribeiro, M., Czekster, R., & Webber, T., 2006. Improving productivity of local software development teams in a global software development environment. *2006 IEEE International Conference on Global Software Engineering (ICGSE '06)*, pp. 253–254. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031773.

Robinson, S. (1997) Simulation model verification and validation: Increasing the users' confidence. In S. Andradottir, K. J. Healy, D. H. Withers and B. L. Nelson, *Proceedings of the 1997 Winter Simulation Conference,* pp. 53-59. Society for Computer Simulation, San Diego, California.

Robinson, M. & Kalakota, R., 2004. *Offshore Outsourcing: Business models, ROI and best practices.* Alpharetta, GA: Mivar.

Robinson, S., 2007. *Simulation: The Practice of Model Development and Use*.West Sussex: John Wiley.

Rodríguez, J., Ebert, C., & Vizcaino, A., 2010. Technologies and tools for distributed teams. *IEEE Software*, 27(5), 10–14.

Rossman, G. & Rallis, S., 2012. *Learning in the Field: An introduction to qualitative research* 3rd edm. Thousand Oaks, CA: Sage.

Runeson, P. & Höst, M., 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 131–164.

Saunders, M., Lewis, P., & Thornhill, A., 2009. *Research Methods for Business Students,* 5th edn. Harlow: Pearson Education.

Sawyer, S., 2001. A market-based perspective on information systems development. Communications of the ACM, 44(11), 97–102.

Sawyer, S., 2000. Packaged software: Implications of the differences from custom approaches to software development. *European Journal of Information Systems*, 9, 47–58.

Schmid, K., 2014. Challenges and solutions in global requirements engineering – A literature survey. *In Proceedings of SWQD (Vienna, Austria)*, pp. 85–99.

Schneider, S., Torkar, R., & Gorschek, T., 2012. Solutions in global software engineering: A systematic literature review. *International Journal of Information Management*, 33(1), 119–132.

Schniederjans, M. J., Schniederjans, A. M. & Schniederjans, D. G., 2005. *Outsourcing and Insourcing in an International Context*. New York: M. E. Sharpe.

Schwaber, K., 2004. *Agile Project Managment with Scrum*. Washington: Microsoft.

References

Schwaber, K. & Beedle, M., 2002. *Agile Software Development with Scrum.* Upper Saddle River: Prentice-Hall.

Sengupta, B., Chandra, S., & Sinha, V., 2006. A research agenda for distributed software development. *Proceeding of 28th International Conference on Software Engineering - ICSE '06*, p. 731.

Setamanit, S., Wakeland, W., & Raffo, D., 2006. Planning and improving global software development process using simulation. *Proceedings of 2006 International Workshop on Global Software Development for the Practitioner - GSD '06*, p. 8. Available at: http://portal.acm.org/citation.cfm?doid=1138506.1138510.

Setamanit, S., Wakeland, W., & Raffo, D., 2007. Using simulation to evaluate global software development task allocation strategies. *Software Process: Improvement and Practice*, 12, 491–503.

Shao, B. B. B. M. & David, J. S., 2007. The impact of offshore outsourcing on IT workers in developed countries. *Communications of the ACM*, 50(2), 89–94.

Sharp, H., Finkelstein, A., & Galal, G., 1999. Stakeholder identification in the requirements engineering process. *Proceedings of 10th International Workshop on Database and Expert Systems Applications. DEXA 99*.

Shore, J. & Warden, S., 2007. *The Art of Agile Development*. Sebastopol: O'Reilly.

Šmite, D., Wohlina, W., Aurumc, A., Jabangwe, R., & Numminen, E., 2013. Offshore insourcing in software development: Structuring the decision-making process. *Journal of Systems and Software*, 86(4), 1054-1067. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0164121212002816 [Accessed December 3, 2013].

Šmite, D. & Borzovs, J., 2006. A framework for overcoming supplier related threats in global projects. *Software Process Improvement*, 4257, 50–61.

Sokolowski, J. A. & Banks, C. M., 2009. *Principles of Modeling and Simulation: A multidisciplinary approach.* New Jersey: John Wiley.

Solingen, R. van & Berghout, E., 1999. *The Goal/Question/Metric Method: A practical guide for quality improvement of software development*. Berkshire: McGraw-Hill.

Sommerville, I. (ed.), 2001. *Software Engineering* 6th edn. Essex: Pearson Education.

Sommerville, I. & Sawyer, P., 1997. *Requirements Engineering: A good practice guide*. Chichester: John Wiley.

Stake, R., 2000. The case study method in social inquiry. In R. Gomm, M. Hammersley & P. Foster (eds), *Case Study Method: Key issues, key texts*, p. 276. London: Sage.

Sulfaro, M., Marchesi, M., & Pinna, S., 2007. Agile practices in a large organization: The experience of Poste Italiane. In *Agile Processes in Software Engineering and Extreme Programming, Proceedings*, pp. 219–221.

Sureshchandra, K. & Shrinivasavadhani, J., 2008. Adopting Agile in distributed development. *2008 IEEE International Conference on Global Software Engineering*, pp. 217–221. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4638670 [Accessed August 17, 2011].

Teruel, M. A., Navarro, E., López-Jaquero, V., Montero, F., Jaen, J., & González, P., 2014. A CSCW requirements engineering CASE tool: Development and usability evaluation. *Information and Software Technology*, 56, 922–949.

Therrien, E., 2008. Overcoming the challenges of building a distributed Agile organization. *Agile 2008 Conference*, pp. 368–372.

Tidd, J. & Izumimoto, Y., 2002. Knowledge exchange and learning through international joint ventures: An Anglo-Japanese experience. *Technovation*, 22, 137–145.

Tiwana, A. & Keil, M., 2004. The one-minute risk assessment tool. *Communications of the ACM*, 47, 73–77.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E., 2011. The Agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology*, 53(1), 58–70.

Weissberg, R. & Buker, S., 1990. *Writing up Research*. New Jersey: Prentice-Hall.

Xu, L. & Brinkkemper, S., 2007. Concepts of product software. *European Journal of Information Systems*, 16(5), 531–541.

Xu, L. & Brinkkemper, S., 2005. Concepts of product software: Paving the road for urgently needed research. In *CAiSE Workshops*, *1st International Workshop on Philosophical Foundations of Information Systems Engineering*, LNCS, Springer-Verlag, pp. 523–528.

Zhang, Y. & Harman, M., 2010. Search based optimization of requirements interaction management. In *Proceeding, 2nd International Symposium on Search Based Software Engineering, SSBSE 2010*, pp. 47–56.

Appendix A : Ethical Approval

Appendix B :

B.1 Questionnaire for the preliminary study

Demographic Information

The Main Goals of This Survey

This questionnaire aims to validate a proposed framework (Fig. 1) to use agile methods to customise software products across organisational boundaries in distributed development projects.

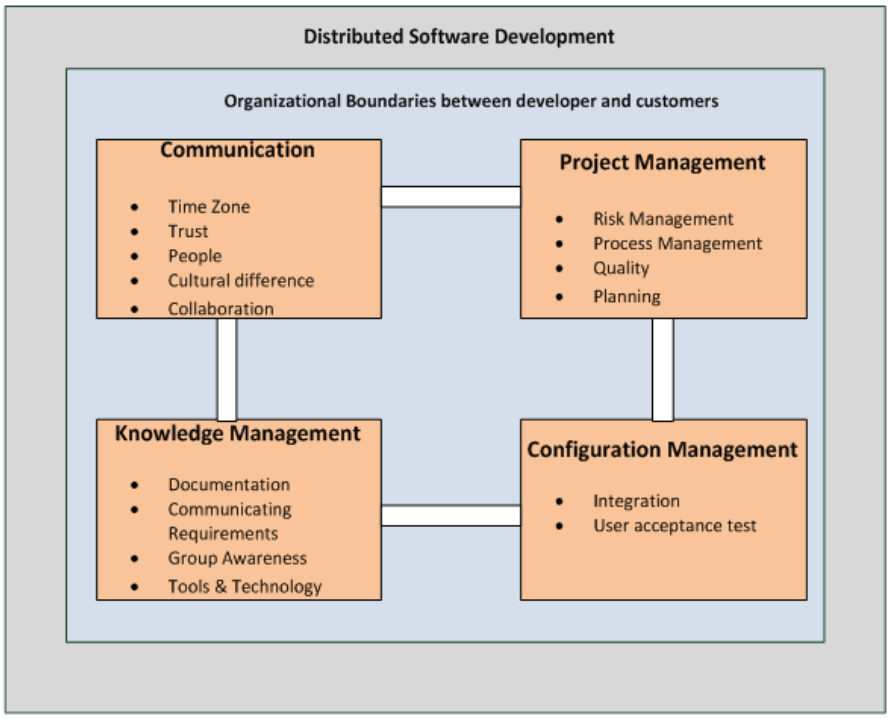


Figure 1: The proposed framework

1) Which of the following positions have you held? *You can select more than one.*

[ ] A software developer

[ ] A system analyst

[ ] A project manager

[ ] A university researcher

[ ] A software tester

[ ] Other (Please write it in the comment box below)

Comments:

_____

_____

_____

_____

2) How long have you been working in software development?*

( ) Less than 2 years

( ) 2–3 years

( ) 3–5 years

( ) More than 5 years

3) For how long have you been using agile methods?*

( ) None

( ) Less than 2 years

( ) 2-3 years

( ) More than 3 years

4) Which type of development projects have you worked on? *You can select more than one.*

[ ] Distributed projects using an agile method (e.g., Scrum)

[ ] Distributed projects using another software development approach (e.g., Waterfall model)

[ ] Collocated projects using an agile method

[ ] Collocated projects using another software development approach

[ ] Other (Please write it in the comment box below)

Comments:

_____

_____

_____

_____

5) How long have you been involved in customisation projects (customising a software code based on customer requirements)?*

( ) None

( ) Less than 2 years

( ) 2–3 years

( ) More than 3 years

The Research Domain

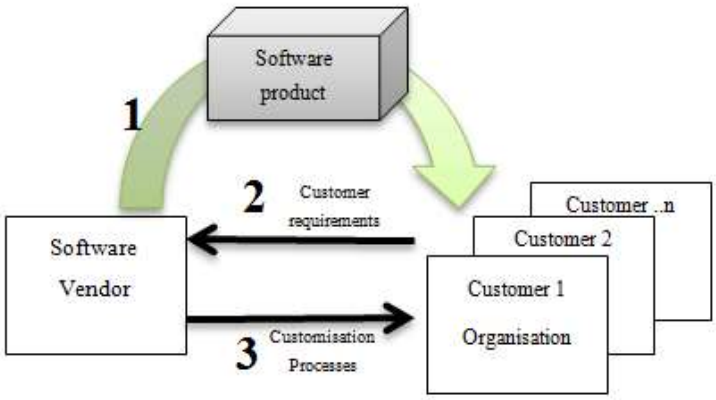


Figure 2: Business model between vendor and customers

6) In the industry, customisation of software code levels is common. In this case shown in (Fig. 2), software developers provide one software product for different customers, then customise it based on customer requirements.*

( ) Strongly Agree

( ) Agree

( ) Neutral

( ) Disagree

( ) Strongly Disagree

7) Agile software development has been successful over the past two decades; however, applying agile software development to distributed development projects still suffers from challenges.*

( ) Strongly Agree

( ) Agree

( ) Neutral

( ) Disagree

( ) Strongly Disagree

8) Organisational (cultural) boundaries affect the development and customisation processes between organisations in distributed software development projects.*

( ) Strongly Agree

( ) Agree

( ) Neutral

( ) Disagree

( ) Strongly Disagree

9) Customisation processes on software codes in distributed development projects need frameworks to address the issues of using agile methods across organisational boundaries.*

( ) Strongly Agree

( ) Agree

( ) Neutral

( ) Disagree

( ) Strongly Disagree

10) Current frameworks are lacking in accommodating the challenges of applying agile methods to distributed software projects for customisation software products across organisational cultures.*

( ) Strongly Agree

( ) Agree

( ) Neutral

( ) Disagree

( ) Strongly Disagree

The Proposed Framework

The questions from 11 to 14 are scaled from 1 to 5, in which 1 means weak and 5 means strong.

( ) 1 (Weak)

( ) 2

( ) 3

( ) 4

( ) 5 (Strong)

14) The framework addresses the challenges and issues associated with using agile software development across organisational boundaries in distributed software development*

( ) 1 (Weak)

( ) 2

( ) 3

( ) 4

( ) 5 (Strong

15) The proposed framework would help development team members in their work. *

( ) 1 (Weak)

( ) 2

Appendix B

( ) 3

( ) 4

( ) 5 (Strong)

Framework Parts

16) Communication is a challenge in distributed software development (DSD) projects. Many aspects related to communication affect the challenges of using agile development in DSD. How strongly are these factors related to communication problems?*

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Time Zone	( )	( )	( )	( )	( )
Trust	( )	( )	( )	( )	( )
People	( )	( )	( )	( )	( )
Cultural difference	( )	( )	( )	( )	( )
Collaboration	( )	( )	( )	( )	( )

17) These factors related to project management bring challenges to distributed software development using agile methods across organisational boundaries. In addition, they affect development processes differently across distributed development teams. How strongly are these factors related to project managment challenges?*

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Risk Management	( )	( )	( )	( )	( )
Process Management	( )	( )	( )	( )	( )
Quality	( )	( )	( )	( )	( )
Planning	( )	( )	( )	( )	( )

18) During the development process in distributed projects, there is a great deal of information and knowledge being transferred. These factors relieve the challenges of knowledge management for this type of development. How strongly are these factors related to knowledgement managment challenges?*

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Documentation	( )	( )	( )	( )	( )
Communicating Requirements	( )	( )	( )	( )	( )
Group Awareness	( )	( )	( )	( )	( )
Tools & Technology	( )	( )	( )	( )	( )

19) These factors represent the interaction between development teams and distributed customers in integrating and deploying the delivered software tested by customers. How strongly are these factors related to configuration management problems?*

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Integration	( )	( )	( )	( )	( )
User Acceptance Test	( )	( )	( )	( )	( )

To improve the framework

20) Which factors should be in this proposed framework?

1-: _____

2-: _____

3-: _____

4-: _____

5-: _____

21) Which factors should not be in this framework?

1-: _____

2-: _____

3-: _____

4-: _____

.22) We are going to conduct interviews with some software development experts and agile practitioners to discuss that framework. Would you like to be involved in these interviews? It would takes about 30 minutes.

( ) Yes, I would like to involve in the interviews. (Please write your e-mail address in the comment box below)

E-mail address:

( ) No, I don't like to involve in this interviews

Thank You!

Thank you for taking our survey. Your response is very important to us.

B.2 The contextual review questions

Reference Number: ERGO/FoPSE/8900	**Version:** 1.0	**Date:** 2014-02-01
Name of Investigator(s): Abdulrahman M. Qahtani		
Title of Study: Customising Software Products across Organisational Boundaries in Distributed Software Development Domains		

Questions

1. How are the company's teams structured?
2. What processes do the development, customisation, and support services follow?
3. How requirements create at customer's location?
4. How are customers' requirements collected for distributed domains?
5. How do you communicate with your customers and your representatives at customer locations?
6. How are decisions made regarding customer requirements?
7. What is the customisation process from issue creation to issue closure?
8. What challenges do you face in terms of communications requirements in distributed domains?
9. Do you have more information about the current customisation process?

Note: The interview questions will be open in order to collect as much as information as possible about the customisation process, current projects processes, and information related to decision-making for the customisation requirements of distributed customers.

B.3 Interview questions for the confirmatory study

1- Confirm the current model

a. **To what extent does the simulation model of the current model reflect the real system of the customisation process in your company?**

b. From the analysis of the historical data of the customisation process for customer's requests, the average waiting time before was as follows:

   i. Decision-making: 5.44 hours

   ii. Approval process: 44.12 hours

   iii. Development process: 160.67 hours

   How much do you think these numbers reflect the waiting time in the real system?

2- Confirm the challenges of communication in the current model

From the contextual inquiry and analysis of the historical data from your system, we have highlighted some implications of the central decision-making in the current model. They are

i. A delay in the rejection and holding process in order to carry out further investigation to understand the customer's requests and discuss them with him/her directly or with the implementer at the customer's location.

ii. A long waiting time before the decision-making process because of the number of requests coming from widely distributed customers.

iii. A delay in the entire customisation process for customers' requests as a result of a long waiting time before both the decision-making and development processes.

What do you think about these implications for central decision-making?

3- Take their opinion on the results of the proposed model

a. Local decision

The proposed model allocates the decision making to the customer's location in order to reduce the challenges of distributed communication and its impacts on decision-making time and the entire customisation process. The following table shows the results of the simulation of the current model and the proposed model.

Factor	Current model (in hour)	Proposed model (in hour)
Decision time	14.70	6.20
Customisation time	191.89	167.80

Considering the evaluation process of the proposed model against the current model, do you think allocating the decision making to the customer's location would decrease the implications of distributed communication for customer's requests on decision time and the time of the entire customisation process?

b. Local decision and development for some requests

The proposed model allocates the decision making to the customer's location and the development process for the requests takes less than 24 hours thereby reducing the challenges of distributed communication and its impacts on decision-making time, development time and the entire customisation process. The following table shows the results of the simulation of the current model and the proposed model.

Factor	Current model (in hour)	Proposed model (in hour)
Decision time	14.70	3.79
Development time	161.18	119.13
Entire Customisation time	189.45	122.25

Considering the evaluation process of the proposed model against the current model, do you think allocating the decision making and development process for some requests to the customer's location would decrease the implications of distributed communication for customer's requests on decision time, development time and time of entire customisation process?

4- General opinion and thoughts

i. **Is there anything else you would like to add before we end?**