Packaged Software Implementation Requirements Engineering by Small Software Enterprises: An Ethnographic Study

Issam Jebreen

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

2014

School of Computer and Mathematical Sciences
Primary Supervisor: Dr. Robert Wellington
Secondary Supervisor: Professor Stephen MacDonell

Table of Contents

List of Figures

LIST OF TABLES

List of Abbreviations Used

BP	Business Process
COTS	Commercial Off-The-Shelf
CRM	Customer Relationship Management
ERP	Enterprise Resource Planning
ICT	Information and Communication Technology
IS	Information Systems
IT	Information Technology
MIS	Management and Information Systems
ORACLE	Company Name
PhD	Doctor of Philosophy
QA	Quality Assurance
RE	Requirements Engineering
SAP	Company Name
SCM	Supply Chain Management
PS	Packaged Software
PSI	Packaged Software Implementation
PSIRE	Packaged Software Implementation Requirements Engineering
SMEs	Small – Medium Enterprises
SMSSDCs	Small – Medium Sized Software Development Companies
VSSE	Very Small Software Companies
SPSVs	Small Packaged Software Vendors

Attestation of Authorship

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

Issam Jebreen

Acknowledgements

This PhD thesis represents the product of an evolving process of learning. To this end, I am grateful to the many people who, directly or indirectly, participated in this process.

I would like to thank Steve MacDonell and Robert Wellington, my supervisors, whose expert guidance enabled me to face the various challenges that emerged as this PhD progressed. I would like to thank Judith Symonds who was my primary supervisor for the first year for her support and help. I would like also to thank Heidi Logan from Auckland University English department for her support and help.

Ethical approval for this research was obtained from the Auckland University of Technology Ethics Committee. The research method was approved on 7 Feb 2011 (reference number 10/293).

I would like to acknowledge the various Jordanian organisations who participated in this study. I am grateful to the organisations involved in the ethnographic study that forms a substantial component of this PhD.

This thesis would not have been possible without the support of my family. I am deeply grateful to my beloved father Yousef Jebreen, to my beloved mother Fatimah Al Waked who supports me in all of my life. I would like also to thank my dear brothers Byder, Zaid, Yazeed, Yazen, Omer, Ahmad, and all of my family for their assistance and support. I am most grateful to all of them for their continuous support and help.

Finally, special thanks to all my colleagues and friends in the Faculty of SCMS, who provided assistance to me in order to complete this thesis.

Abstract

This thesis investigates the practices of requirements engineering (RE) for packaged software implementation, as enacted by small packaged software vendors (SPSVs). Throughout the thesis, a focus on the actions carried out by SPSVs analysts during RE is maintained, rather than a focus on the actions of client companies. The thesis confirms the assertions literatures finding that most contemporary RE practices are unsuitable for SPSVs. The research investigated the means by which SPSVs can adopt, follow and adapt the best possible RE practices for packaged software implementation (PSI), through the collection of qualitative and quantitative data during an ethnographic study in two small- to medium-sized software development companies in Jordan. Through discussion and analysis of the RE processes witnessed during the ethnographic study, the thesis answers the research question "What are the analysts' practices in the context of packaged software implementation by small packaged software vendors?", it provides a comparative analysis of the practices used in traditional RE and in packaged software implementation RE (PSIRE), and in doing so it also identifies novel RE practices for packaged software implementation that have not been noted in prior research. Contribution is made to the understanding of RE practices as related to packaged software implementation through the in-depth discussion of innovative RE practices and particularly through a thorough explanation of the feasibility study for PSI. The research findings lead to the provision of a Parallel Star Model depicting the practices of packaged software implementation RE at SPSVs. This Parallel Star Model demonstrates that more than one RE practice may be carried out at the same time during packaged software implementation, and provides guidance for those SPSVs about to engage in RE for packaged software implementation.

This study also delivers an innovative understanding of PSIRE: first, the analyst has more of a hybrid analyst-sales-marketing role. Secondly, the use of a live scenario software demonstration in order to convince the client to buy into the PS may lead to increased perceived feasibility and reduced resistance to PS implementation. Thirdly, the assessment criteria that are used to estimate the effort and time needed for PS implementation are new features, level of customisation, software 'output', and technical needs. Fourthly, the Parallel Star Model shows that during PSIRE, more than one RE process can be carried out at the same time. The Parallel Star Model has few constraints, because not only can processes be carried out in parallel, but they do not always have to be followed in a particular order. This study therefore offers a novel investigation and explanation of PSIRE practices, approaching the phenomenon from the viewpoint of the analysts, and offers the first extensive study of packaged software implementation RE (PSIRE) in SPSVs.

List of Research Outputs

Jebreen, I., Wellington, R., & MacDonell, S. G. (2013, 2-5 December). Packaged Software Implementation Requirements Engineering by Small Software Enterprises. Paper presented at the 20th Asia-Pacific Software Engineering Conference (APSEC 2013), Bangkok, Thailand.

Jebreen, I., R. Wellington, and S. MacDonell. *Understanding Feasibility Study Approach for Packaged Software Implementation by SMEs*, in *22nd International Conference on Information System Development* 2013. Seville, Spain.

Jebreen, I. and R. Wellington, *Understanding Requirements Engineering Practices for Packaged Software Implementation*, in *4th IEEE International Conference on Software Engineering and Service Sciences* 2013: Beijing, China.

Jebreen, I. and R. Wellington, *Packaged software implementation as requirements engineering practices at SMSDCs.* American Academic & Scholarly Research Journal 2013. **5**(3).

Jebreen, I., *Explore Developers-Users Communication: Using Grounded Theory Method to Investigate Usage of Nonverbal Channels in Requirement Determination Phase*, in *Australasian Conference on Information Systems 2010, Doctoral Consortium 2010:* Brisbane, Australia.

Chapter 1 Introduction

Things should be made as simple as possible, but no simpler.
[Albert Einstein]

1.1. Introduction

In recent years the market in which packaged software (PS) is sold to large companies has become saturated (Morabito et al., 2005). PS companies and vendors have therefore begun to target the small to medium-sized PS market (attempting to sell packages to small to medium-sized enterprises (SMEs), and various midrange or less complex software packages have been developed (Zach et al., 2012). SMEs are of critical importance to many economies. According to Snider et al. (2009), SME firms "with less than 500 employees provided 51 per cent of all employment in the USA as of March, 2004 and 64 per cent of all Canadian private sector employment in 2005. In the European Union, firms with 250 employees or less provided 67 per cent of employment in 2003". While SMEs are an integral part of economies, they face managerial challenges implementing packaged software (Snider et al., 2009; Haddara & Zach, 2011; Zach et al., 2012).

SMEs are considered to be fundamentally different from large enterprises (LEs) in several respects (Laukkanen et al., 2007). Some distinguishing characteristics of SMEs include ownership type, culture, structure, and market orientation (Ghobadian & Gallear, 1997; Wong & Aspinwall, 2004). Other researchers have found that when it comes to IT/IS adoption, SMEs are constrained by limited resources and limited IS knowledge, or by a lack of IT expertise (Buonanno et al., 2005; Laukannen et al., 2007). Studies of PS implementations have argued that findings about implementations in large companies cannot be applied to SMEs (Haddara & Zach, 2011; Zach et al., 2012). These distinguishing characteristics of SMEs may influence the PS implementation issues they face (Zach et al., 2012). PS implementation remains a challenge for many SMEs (Malhotra & Temponi, 2010; Olson & Staley, 2012; Zach et al., 2012). Despite the importance of PS implementation being recognised by former studies, there has been little research exploring this issue further. In particular, discussions about SMEs rarely occur in the literature about PS implementation, and how the structure of SMEs shapes the software throughout its life cycle of implementation is rarely mentioned (Haddara & Zach, 2011; Zach et al., 2012).

SMEs generally thrive because they have successfully done something unique within a niche market (Newman & Zhao, 2008; Zach et al., 2012). For this reason, SMEs may seek to protect their competitive advantage by avoiding any standardisation encouraged by packaged software (PS) (Buonanno et al., 2005; Laukannen et al., 2007; Newman & Zhao, 2008; Zach et al., 2012). Packaged software implementation at SMEs therefore involves challenges related to how best to respond to misalignments between the functionality offered by packaged software and the business needs/environment of SMEs (Light, 2005; Zach et al., 2012). This challenge is one of the key issues in packaged software implementation at SMEs. One question related to this issue is which processes Small packaged software vendors (SPSVs) apply in order to identify misalignments between packaged software functionality and client requirements and how they deal with these misalignments. Implementation consists of the customisation, installation, configuration and adaptation of the packaged software acquired according to the needs of the organisation, and a better gap/fit analysis between the organisation's needs and the functionality of packaged software can be achieved through Requirements Engineering (RE).

However, several software engineering researchers have argued that most current requirements engineering practices are unsuitable for SPSVs (Aranda et al., 2007; Bürsner & Merten, 2010; Jantunen, 2010; Quispe et al., 2010; Merten et al., 2011) and that SPSVs are unable to successfully apply RE methods and techniques that have been designed for larger companies (Bürsner & Merten, 2010; Merten et al., 2011). For this reason, research into RE practices should focus more on the investigation of how RE is practiced in smaller companies such as SPSVs and on how the RE methods followed by SPSVs can be improved. This thesis therefore features an ethnographic study of how the RE of packaged software implementation is enacted in Small to Medium-sized

Software Development Companies. In this study, packaged software is defined as ready-made software products that generally require modification or customisation for specific organisations. They are often exemplified by enterprise resource planning (ERP) (Xu & Brinkkemper, 2005). The RE related to Packaged Software Implementation will hereafter be referred to as PSIRE throughout this thesis.

1.2. Rationale of the Study

The poor use of requirements engineering (RE) practices has often been identified as one of the major factors that can jeopardise the success of a software project (El Emam & Madhavji, 1995; Aranda et al., 2007). Meanwhile, researchers have also recognised that following appropriate RE practices contributes to the success of software projects (Solemon et al., 2009). For example, Aranda et al. (2007) state that gathering and managing requirements properly are key factors when it comes to the success of a software project. There is a general critical consensus that RE practices play a very important role in the success or failure of software projects (Merten et al., 2011). However, it is not possible to improve RE practices until areas that need improvement in an organisation's current RE practice have been identified (Bürsner & Merten, 2010; Quispe et al., 2010). Meanwhile, the solution for improving RE practices will be different in each company; it has been found that a one-size-fits-all approach does not work in such a scenario (Nikula et al., 2000; Aranda et al., 2007; Cox, 2009; Bürsner & Merten, 2010; Quispe et al., 2010).

Merten et al. (2011) argue that previous findings suggest that SPSVs may not actually need to have extremely formal and conventional forms of RE. Instead, "light organisation and unconventional RE" may work better for many SPSVs. They also discuss the various RE models that have been provided in previous studies. For example, Olsson et al. (2005) presented a pragmatic framework for RE in SPSVs. However, Merten et al. (2011) suggests that the list provided needs to be expanded in future because the selection of RE techniques is a central problem in all aspects of process improvement. They note another study by Hardiman (2002) but observe that the

RE practices and techniques discussed in Hardiman's study often seem to be specifically tailored toward particular individual SPSVs and therefore do not seem to offer solutions that can be applied to the whole domain. Pino et al. (2008) provide an extensive list of Software Processes Improvement (SPI) models. The methods Pino et al. (2008) discuss are based on ideas put forward by the Software Engineering Institute (SEI) or by the International Organisation for Standardisation (ISO). However, Pino et al. note that many of the models proposed by these two organisations could be too complex for SPSVs to apply.

Furthermore, the conclusion of the 1st Workshop on Requirements Engineering in Small Companies (2010) was that "existing RE techniques are not sufficient for small companies" and that some other factors need to be realised, such as that size is "not the only measure to categorise smaller companies and describe the exact focus of research", "that tacit knowledge and social structures" in place in SMEs [SPSVs] may play an important role in RE research, that introducing RE methods designed for larger companies may actually harm the specific features of an SME [SPSVs], and that RE methodologies need to be made more lightweight.

Therefore, this study presumes that the specific characteristics of SPSVs, SMEs and packaged software may influence RE. The recent literature has paid little attention to RE of PS implementation from the perspective of SPSVs (Jantunen, 2010; Zach et al., 2012). In many cases software engineering no longer involves building systems from scratch (Sommerville et al., 2012), but rather integrating "existing frameworks and modules" or working with a "comprehensive code base" (Dittrich et al., 2009). Software engineering has a group of influential approaches that are often considered good practice. These involve such practices as "structured programming", "stepwise refinement", and collecting "a complete set of test cases" (Dittrich et al., 2009), but these approaches don't apply for PS implementation. Dittrich et al. (2009) argues that such implementation requires "independent consideration". In this study, therefore, I set out to discover which RE practices are actually being used in RE by SPSVs.

1.3. Problem Statement & Research Questions

Often, SPSVs are unable to apply RE methods and techniques without modifications (Aranda et al., 2007; Bürsner & Merten, 2010; Merten et al., 2011). In addition, shortcomings in applying RE methods due to time constraints or limited resources may arise (Aranda et al., 2007; Bürsner & Merten, 2010). Therefore, "RE research has to intensify the investigation of RE practices in SMEs [SPSVs]. Otherwise SMEs [SPSVs] will have to continue their search for methodical orientation and dedicated tool support. Normally, the people responsible for requirements in SMEs [SPSVs] are ambitious, but suffer from scarcity of resources. Their time for doing experiments and trying different methods is very limited. They need quick methodical improvement of requirements elicitation, documentation, communication and traceability as well as more continuity of requirements management through the whole software lifecycle" (Bürsner & Merten, 2010, p137). According to past literature, there is a clear problem related to whether SPSVs can carry out effective RE. Figure 1.3-1 shows the research area of this study.

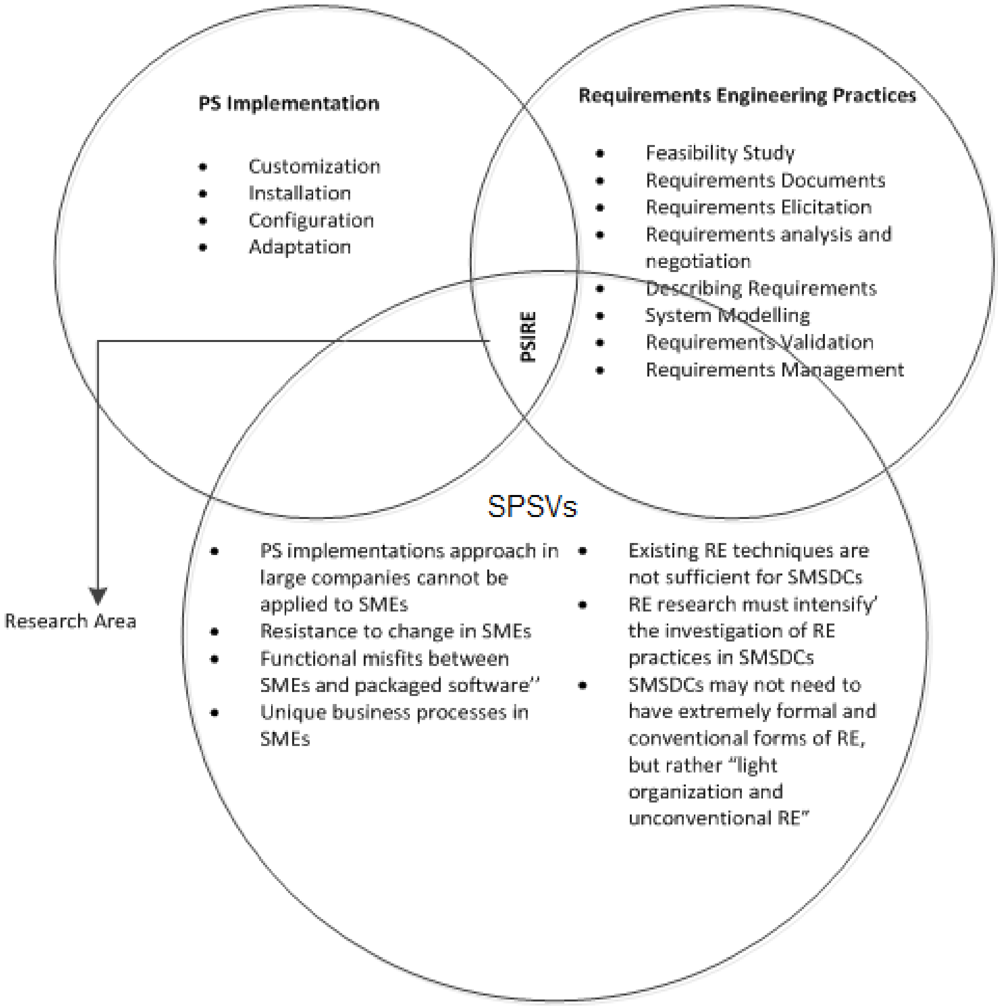

PS Implementation
Customization
Installation
Configuration
Adaptation
Requirements Engineering Practices
Feasibility Study
Requirements Documents
Requirements Elicitation
Requirements analysis and negotiation
Describing Requirements
System Modelling
Requirements Validation
Requirements Management
PSIRE
Research Area
SPSVs
PS implementations approach in large companies cannot be applied to SMEs
Resistance to change in SMEs
Functional misfits between SMEs and packaged software"
Unique business processes in SMEs
Existing RE techniques are not sufficient for SMSDCs
RE research must intensify' the investigation of RE practices in SMSDCs
SMSDCs may not need to have extremely formal and conventional forms of RE, but rather "light organization and unconventional RE"

Figure 1.3-1 Research area

Meanwhile, Karlsson et al. (2007) state that there are "several studies that concern or include RE issues. However, none of these focus primarily on packaged software development and implementation. Furthermore, in most of these studies, the studied projects and organisations are mainly large, both in terms of the number of persons and requirements involved, and in terms of the duration of the projects". Quispe et al. (2010) highlight that "there is a lack of knowledge about the requirements engineering practices in these types of companies [small-medium]". Researchers in the field encounter a general lack of information when it comes to gaining knowledge about how RE is carried out in packaged software companies. It is in fact difficult for researchers to gain much knowledge about how SPSVs carry out RE given that most SPSVs seldom request external support, probably due to limited finances. However, RE research should eventually enable those companies to become aware of more state of the art or innovative RE techniques and to be able to improve their RE practice without external help (Merten et al., 2011).

One core question that remains, despite the work done in previous studies, is: how is the RE of packaged software implementation enacted in small packaged software vendors (SPSVs)? While focusing on this question I shall construct a "theory for explaining/understanding" to represent how and why events occur during the implementation process of PS in terms of RE. According to Gregor (2006, 2007) this type of theory is suitable when the researcher uses an interpretive paradigm. As this form of theory is formulated in order to aid the understanding of phenomena, making testable predictions about the future is not the primary concern of this research. This study set out to understand packaged software implementation (customisation, installation, configuration and adaptation) in terms of RE practice at SPSVs. In the context of packaged software implementation by small packaged software vendors, the research questions that shall be answered through this study are:

- What are the analysts' practices?
- How do analysts conduct these practices?
- Why do analysts conduct these practices?

1.4. Aims of the Study

The purpose of this study is to provide an understanding of the phenomenon of Packaged Software Implementation in terms of how it interacts with requirements engineering practices. The reason for approaching the topic this way is that since Packaged Software Implementation involves activities such as installation configuration, and modification of packaged software products, and requires analysts to find the needs of the client organisation, one of the main tasks of an analyst is to be able to identify requirements and misfits. Therefore, this study shall explore and investigate analysts' requirements engineering practices including their activities, strategies and actions in undertaking PS implementation. This study sets out to achieve the following specific aims:

- To present a Small - Medium Sized Software Development Companies' practice viewpoint on packaged software implementation in terms of RE; this viewpoint has been neglected in the literature.
- To provide a comprehensive understanding of how work is organised in such settings.

1.5. Scope of the Study

This study focuses on exploring the RE practices used by SPSVs in terms of RE for PS implementation. RE is defined as "a systematic approach to eliciting, organising, and documenting the requirements of the system and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system" (Leffingwell & Widrig, 2003). This study uses the term "requirements" in the same way used by Soh et al. (2000) & Davis et al. (1993). Davis et al. (1993) state that a requirement is "a user need or a necessary feature, function, or attribute of a system that can be sensed from a position external to that system" and Soh et al. (2000) state that misalignments (also referred to as 'misfits' in the literature) can consist of "the incompatibilities between organizational requirements and packaged software in terms of data, process, and output". The term 'users' refers to those who use and interact with the system to get work done.

The concept of PS is defined as a ready-made software product developed by software companies (vendors of packaged software and software houses). The product can then be obtained from software companies. The product generally requires modification or customization for specific markets. They are often exemplified by enterprise resource planning (ERP) systems.

PSIRE is a systematic approach of analysts' practices which focuses on demonstrating what function the software provides and who needs the particular function in order to do their job, and on identifying what misalignments exist between software functions and user needs. All of this could be done by using live scenario software demonstrations and by considering specific assessment criteria when making decisions about misalignments. PSIRE actually starts at the pre-implementation process and it continues during both the pre-implementation phase and the implementation phase. In the context of PSIRE, analysts do not just collect user needs, they also demonstrate software, train users, find misalignments, and figure out how to deal with the misalignments.

PS implementation involves several activities such as customisation, installation, configuration and adaptation. In the PS acquisition context, “detailed analysis comes after purchase. It is only during installation that users become deeply involved for the first time in assessing how the software meets their needs” (Sawyer, 2001). According to Sawyer (2001) the “installation phase involves many detailed analysis issues”. Configuration refers to the non-functional customisation of the PS (Nordheim & Päivärinta, 2004). Adaptation refers to two scenarios: fitting organisational processes to the software, or customising the software to fit the processes (Chiasson & Green, 2007). Customisation in this study refers to changing or modifying packaged software in terms of interface, transactions, and functions (Light, 2005; Dittrich et al., 2009).

According to the 2011 United Nations Development Programme’s “Jordan Human Development Report” which quotes the 2011 “Small Business and Human Development” report detailing business development in Jordan, “Micro, Small, and Medium-sized Enterprises (MSMEs) account for 37% of total employed and 60% of total employment in the private sector in Jordan” (UNDP, 2011). The “Small Business and Human Development Report” states that “MSMEs comprise 99.6% of all firms outside the agricultural sector”. The Technical Committee engaged for the report, “made up of relevant experts from across Jordan defined small enterprises as those with less than 20 employees, and medium enterprises as those with between 20 to 99 employees”. Therefore, the organizations focused on in this study match up with the size of the small Jordanian enterprises mentioned in the above report.

This study focuses on how SPSVs interact with ‘local users’ - here, this term refers to the client organisations that the SPSVs do business with. In this study, the local users were small organisations (SMEs) doing business in the same country as the SPSVs. Due to this focus, the business globalisation of the SPSVs is not considered. This study also focuses on SPSVs as packaged software developers and vendors of packaged software.

1.6. Contributions

If the stated research aims of this study are fulfilled, the practice of packaged software implementation may be enhanced. The study will provide a better understanding of PSIRE in SPSVs, and any improved effectiveness of packaged software implementations may benefit both software development companies and key stakeholders. The realisation of the research aims will also create possibilities for future research in areas related to packaged software implementation and requirements engineering. In general terms, the current study shall contribute to the IS body of knowledge in the following way:

- Making an original contribution to the literature and practice in the field of packaged software implementation at SPSVs, by providing an improved understanding of companies’ practices in terms of requirements engineering.

By better explaining PSIRE phenomena, we may gain an improved understanding of requirements engineering for packaged software implementation and be able to deliver knowledge that can benefit software development companies, owners, and other stakeholders.

1.7. Research Approach

This study is conducted from an interpretive perspective, and treats the SPSVS as the unit being studied. Such an approach provides support for conducting a study that captures the views of the participants, and for understanding SPSVs’ practices in terms of requirements engineering (RE) for packaged software (PS) implementation. The approach can be categorised as interpretive research with the goal of empirical investigation of the software companies' practices (Klein & Myers, 2001).

Inductive analysis, as used in this study, refers to an approach that primarily uses detailed reading of raw data to derive concepts, themes, and models through the researcher's interpretations of the raw data. In other words, the researcher begins from the area of study and creates a theory from the collected data (Hammersley & Atkinson, 2007).

The purpose of inductive analysis is to allow results to emerge from the frequent and significant themes discovered in the raw data without imposing any pre-conceived structure. Unlike deductive analysis, where the key topics are usually reframed because a prior hypothesis is imposed on the data in the hope of obtaining a desired result, an inductive approach describes the actual phenomena through collected data. In other words, the collected data is used in a way that may provide insight into phenomena, rather than following a process whereby data collected will be used to support or refute an already decided or desired result (Fossey et al., 2002).

An ethnographic research method was applied in relation to two software development companies who participated in this research. These will be called Organisation 1 and Organisation 2. The business of both organisations considered in this study is dominated by the provision of packaged software solutions. This study follows Hammersley & Atkinson's (2007) and Schultze's (2000) discussions of various features of the ethnographic research method.

Hammersley & Atkinson's (2007) method for ethnographic research emphasises the following:

- People's actions are studied in their everyday context, rather than under conditions created by the researcher. In other words, the research takes place in the field (p.4).

- Data is gathered from a range of sources, including documentary evidence of various kinds, but participant observation and/or relatively informal conversations are often the main means of collecting data (p.4).
- Data collection is, for the most part, relatively 'unstructured' in two senses. First, it does not involve following through with any specific fixed and detailed research design. Secondly, the categories that are used to interpret what people say or do are not built into the data collection process through the use of observation schedules or a questionnaire. Instead, they are generated through the process of data analysis (p.4).
- The focus is usually on a few cases, generally of a fairly small scale, perhaps a single setting or single group of people. This is to facilitate in-depth study (p.4).
- The analysis of data involves interpretation of the meanings, functions, and consequences of human actions and institutional practices, and how these are implicated in local, and perhaps also wider, contexts (p.4).
- What are produced, for the most part, are verbal descriptions, explanations, and theories; quantification and statistical analysis play a subordinate role (p.4).

Schultze (2000), describing her ethnographic field work within a U.S. Fortune 500 company, explains how she made use of ethnography and attempted to balance objectivity and subjectivity. The company she observed was engaged in an implementation project. Schultze (2000) was in the field for over eight months and occupied two slightly different positions in relation to two different groups of workers within the company. In relation to the company's system administrators and librarians, she was an "active-member-researcher" taking on some work responsibilities, and in relation to analysts within the company she remained a "peripheral-member-researcher" who did not take on work responsibilities.

When collecting data, Schultze (2000) opted for "unstructured interviews and informal conversations" as she felt that these would "not jeopardise my ability to observe in-the-moment reactions and behaviors". She attempted to collect spontaneously given information rather than politically correct answers. When interviewing those people she could only speak with once, however, Schultze (2000) used semi-structured interviews and tape-recorded the conversations. Schultze also took hand-written notes in the field and then re-wrote them more descriptively later in the day.

Schultze notes that the “hallmark of the ethnographic method’ has always been ‘non-interventionist observation”. While Shultze acknowledges that the ethnographer can never avoid some subjectivity creeping into their data collection or interpretation of data, she sought ways to minimise any contamination of data. One way Schultze recommends of minimising such contamination is by “using emic terms, i.e., those indigenous to the social setting”; another is allowing things to happen without interfering.

Schultze also discusses the methods of analysis involved with using qualitative data. Schultze suggests that when analysing field notes and beginning to construct themes inductively the researcher’s own theoretical and personal biases will inevitably play a role in the emergence of such themes. In order to mitigate this, the researcher must juggle induction (“using situated and subjective knowledge”) with deduction – which involves “applying objectified methods, frameworks, and theories to the data”. While applying Schultze’s (2000) method for ethnographic analysis, I shall also use Sommerville & Sawyer (1997) and Cox et al.’s (2009) lists of good practices in RE in order to compare traditional RE practices with practices in PSIRE.

In abstract terms, Schultze’s study describes the work practices of engaging in “expressing”, “monitoring”, and “translating” that are carried out by knowledge workers. These practices are engaged in as a means of balancing the subjective nature of information produced “with activities that enhance[d] its apparent objectivity”. ‘Expressing’ involves “converting subjective knowledge and subjective insights into informational objects that are independent of the knowledge worker”; ‘monitoring’ involves “gathering information in an unobtrusive manner so as to minimise the risk of contaminating the information”; and ‘translating’ involves “creating information by manipulating it and ferrying it across multiple realms until a coherent meaning emerges”.

1.8. Organisation of the Thesis

This thesis consists of eight chapters. The first chapter provides an overview of the background of the study, followed by the problem statement, research questions, significance of the study, scope of the study, and finally organisation of the thesis.

The second chapter provides an extensive review of literature related to the concepts of this study, namely: packaged software, requirement engineering, and packaged software implementation.

The third chapter covers the research approach that was utilised throughout this research. This includes research philosophy, research method, data collection methods, and an overview of data analysis strategies.

The fourth chapter provides a detailed discussion of the data analysis strategies. It starts with an overview of the inductive approach and follows this with a discussion of data analysis strategies.

The fifth and sixth chapters present the results of this study. They start with an overview of the requirement engineering model for packaged software implementation, then move on to reporting about the model elements.

The seventh chapter concludes the study with a recapitulation of the study, an overall discussion of the findings, discussions of the study's implications and of its limitations, and provides suggestions for future study. The eighth chapter then offers a final conclusion of the whole study.

1.9. Summary

As introduced in this chapter, a literature review of previous writing on the topic of RE in Small to Medium-Sized companies reveals that there are various gaps in knowledge relating to RE practices in SPSVs. However, one of the most common conclusions of studies of RE in companies producing packaged software is that RE practices within

SPSVs are different than the RE used in larger companies. Furthermore, most of the studies conducted about PS implementation have considered companies located in America, Australia, Europe, and Asia. There is a shortage of studies investigating SMEs in Africa or in the Middle East. This study uses a conceptualisation of PSI as RE in which a PSI emerges from a dynamic and interactive relationship between the technology, its social and organisational context, and the negotiated actions of various individuals and groups. In general, existing literature has adopted a one sided perspective (in relation to data collection). It has focused on the customer side, whereas other perspectives could enhance the understanding of certain phenomena.

Previous literature on these issues has not addressed the various means software companies adopt when working to implement packaged software. Consequently, there is currently a need to investigate those processes that companies - especially those that are SMEs - apply when managing, implementing, and using packaged software. This need was identified as long ago as 2007, when Light & Sawyer (2007) stated that there was a "need to theorise about packaged software and its place within the field of information systems". In addition, this field has an intrinsic volatility, and according to Xu & Brinkkemper (2007) "despite the economic importance of product software, there is still very limited research activity on the development of software as a product. In the field of product software, academia and companies have not yet developed any satisfactory scientific theory on integrated business models, software development and software implementation". Particularly, "customised development methods have not been addressed yet. These form barriers for product software companies". Dittrich et al. (2009) underline the need to rethink software engineering and programming methods and tools for packaged software. In this study I focus on the practices followed by software companies when implementing packaged software, and by doing so contribute new research to the existing body of knowledge.

Chapter 2 Literature Review

2.1. Introduction

In this chapter I discuss the previous studies that provide the necessary content and background for the research questions I posed earlier about RE in relation to Packaged Software Implementation, and in relation to the more specific topic of Packaged Software Implementation RE as practiced in SPSVs. The discussion of these previous studies, however, is impacted by the study philosophy, in terms of the chronology of when I wrote this chapter. Since this study uses an interpretive philosophy, where I analysed the data inductively, this literature review chapter was written after the chapter in which I stated the data analysis results. However, some examples of the more general literature written on packaged software implementations were considered as I began this study. Once I started to develop the theory that emerged from my research after I had conducted field work and obtained results, I studied literature pertaining to more specific topics in this field, such as packaged software implementation in SMEs and studies of various concepts of implementation. I conducted my research in this order, so that I could, as much as possible, avoid allowing information from the literature review to impact upon the data analysis process.

The purpose of this chapter is to provide insight into the processes involved in packaged software, and into packaged software implementation, and requirements engineering. In this chapter I argue that packaged software implementation relies on a set of activities of software integration, customisation, adoption, interaction between analysts and users, and the identification of misalignments between PS functionality and users' requirements, a process that is applied by the members involved in the implementation. Additionally, in this chapter, I introduce and describe various elements of packaged software implementation. If the efficiency or effectiveness of many of these elements could be improved, this could lead to increased effectiveness within the packaged software implementation process.

2.2. Background

Many organisations have implemented Packaged Software (PS), and the PS market is one of the fastest growing markets in the software industry. A study by AMR research in 2007 predicted that the PS market would grow to $64.88 billion by 2009. PS products are vended onto the open market; such forms of software may be very attractive to potential clients (Staehr et al., 2012).

Software companies moving towards developing packaged software face managerial product development challenges (Gorschek et al., 2012). Sawyer (2000) identified what was then a lack in the degree of research effort devoted to studying the packaged software domain, and sought to address this gap in knowledge by identifying key differences between the development of packaged software and the development of bespoke software. Sawyer identified differences at four levels, which include “industry forces, approaches to software development, work culture, and development team efforts”. His discussion of the differences is augmented by identifying and discussing five different stakeholder groups involved in both bespoke development and PS development. Sawyer (2000) identifies some of the differences between packaged software and bespoke software, stating that “custom IS are those made by either an organisation’s internal staff or by direct subcontract to a software house”. Sawyer goes on to identify ERP software (which he describes as “large packages”) as the fastest growing segment of packaged software. Packaged software systems require extensive tailoring for their implementation, which usually requires the help of consultants, training, and support staff (Sawyer, 2001; Moon, 2007; Wagner et al., 2010; Olson & Staley, 2012). An ERP system is a ‘hybrid’ form of software; even though ERP systems are not built in-house by a company, they usually go through post-implementation tailoring (Gorschek et al., 2012; Sawyer, 2001).

When comparing observable differences between packaged software development and bespoke development, Sawyer suggests that, relative to bespoke software, packaged software development is largely controlled by time pressures rather than cost. In addition, the products made by the packaged software industry have different measurements of success than bespoke products do. The success of packaged software is generally measured in terms of profit, market share, and public awareness of the product. However, with bespoke software, 'success' is often measured by whether the particular company the product was built for is happy with how it functions (Sawyer, 2001). With packaged software development, any concerns of users are filtered through intermediaries, and the development of the software is carried out by developers who hold "line positions" in their software development company (Karlsson et al., 2007; Gorschek et al., 2012).

Sawyer (2000, 2001), Karlsson et al. (2007), and Gorschek et al. (2012) also suggest that with packaged software, there is more of a "product" view of development: the focus is on delivering a particular product that can be sold to many, and the vision for the product is generally that it will evolve "through a planned set of releases". With bespoke software, the focus is more on a "process" view of development. The main conclusion reached by Sawyer (2000, 2001), Karlsson et al. (2007), and Gorschek et al., (2012) is that the development methods for packaged software and bespoke software are very different, and their studies show these differences at the industrial level and through the approaches to developing software, the work culture, and the efforts of the development teams, while also showing the implications that the two approaches hold for various stakeholder groups (Sawyer, 2000, 2001; Karlsson et al., 2007; Gorschek et al., 2012).

Meanwhile, Gorschek et al. (2012) and Karlsson et al. (2007) have pointed out requirements engineering (RE) differences between bespoke and packaged software solutions. In order to suit a wide variety of customers, packaged software is often developed in several consecutive releases. The RE required for packaged software differs from bespoke RE in several ways (Sawyer, 2000). This is due to the fact that packaged software is developed based on potential users, or an imagined group of people who might fit the profile of an intended product client. The fact that packaged software RE elicits requirements based on such a group of potential or imaginary clients is one of the major distinguishing characteristics between packaged and bespoke solution RE (Karlsson et al., 2007).

In recognition of these issues, in recent years extensive research has been conducted to improve the effectiveness of packaged software RE. For example, Karlsson et al. (2007) investigated practices and challenges for PS requirement engineering in Swedish software development organisations in order to increase the understanding of this particular form of requirement engineering. Many of the issues they discovered were unique to PS and not applicable for bespoke software; their results suggest, therefore, that requirements engineering methods that help with the development of bespoke software may not be especially useful in terms of supporting the development of PS. In a later study, Gorschek et al. (2012) found that the requirements engineering models and methods used to develop bespoke software have limited utility when it comes to developing PS. Over the last few years, other studies of the subject have emerged. Barney et al. (2008) studied release planning processes for PS, Lehtola & Kauppinen (2006) investigated the suitability of requirements prioritisation methods for PS, and Mishra & Mishra (2009) suggested a dynamic requirements engineering approach of agile methods for PS.

There are major challenges involved with requirements gathering for creating the initial software for marketing among potential users. However, once potential users are identified, they may be a key for gathering requirements that might potentially fit with the usage of a number of other users with a similar business profile. Nevertheless, the initial requirements are generally invented by the developers themselves, who base their ideas on strategic business objectives, domain knowledge, or a project vision (Karlsson et al., 2007). This makes it reasonable to assume that not all packaged software is suitable for each and every Client's Requirements ('CRs') (Sia & Soh, 2007; Chaisson & Green, 2007). In other words, conflicts might exist between Packaged Software Functionality (PSF) and a client's requirements when software companies implement PS at their client's site. To reduce any negative consequence, efforts must be made to minimise the potential gaps between PSF and CRs (client's requirements) in order for the system to be successfully presented, and accepted and used by the client.

2.3. Concepts of Packaged Software

One previous study by Xu & Brinkkemper (2005) identifies a lack of study devoted to "the engineering of software as a product [packaged software]". They suggest that despite the growing popularity of standard software products, "there is still very limited research activity on the development of software as a product" and "a body of knowledge needs to be established with theories, methods, and tools". Their study reviews terms used in relation to software products and categories of software products, and then discusses elements "of the software business" and a "software product development framework".

Xu & Brinkkemper (2005) make a distinction between "shrink-wrapped software" "commercial off-the-shelf software (COTS)", and "packaged software and commercial software". They note that the distinctions between these products have not always been kept clear in the literature (Xu & Brinkkemper, 2005, 2007). For example, ERP systems belong to the subsection "packaged software" and should not be assigned to any of the other subsections mentioned above (Xu & Brinkkemper, 2007)

Xu & Brinkkemper (2005, 2007) also provide a definition of what a "software product" actually is: "a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market", and they follow this definition by mentioning various 'categories' of software products. They state that according to the OECD (Organisation for Economic Co-operation and Development), software products can be classified as either "system infrastructure software, software development tools, or application software". For example, ERP systems are a kind of "application software". Another means of classification is by viewing the products in relation to "architectural standards and language standards".

Damsgaard & Karlsbjerg (2010) define a software package as “a collection of software components which when combined perform a set of generalised tasks that are applicable to a range of users”. “The core components of a package are identical across all user organisations”, but each organisation can further configure the package to suit their needs. They point out that the choice of a software package may have wide-ranging effects, even on parts of an organisation that do not use that software package, as packages are configured to connect with other software systems within a company.

Similarly, Sawyer (2001) distinguishes a “software product” from an “information system”, noting that an information system may involve the use of more than one software product and also include the hardware and people that help to make the product work. Therefore, buying one piece of software “does not mean the consumer now has an information system”. He suggests that the trends in the development of software have led organisations that consume software to “increasingly assemble pieces, not build them. That is, software consumers focus on ISD [information system development], while vendors focus on developing packaged products”.

Sawyer (2001) briefly discusses the software development market and the trend toward specialisation within various software manufacturers. He also considers subjects such as knowledge transfer and suggests that the activities involved with “requirements analysis and implementation” are geared toward filling “information asymmetries” associated with ISD. He points out that “despite [it] being difficult for developers to build and deliver the kinds of systems users want, it is even more difficult for users to build and install their own systems”.

The goal of an installation of new packaged software is to get the new software operating in a way that allows individual end users to help their organisations improve overall business performance (Sawyer, 2001; Plant & Willcocks, 2007). Such installation is “a project of guided discovery, since identifying risk is difficult in advance”. Sawyer (2001) suggests that with packaged software (market-oriented and market-driven) ISD, detailed analysis of the product by the consumer organisation only comes after purchase. Meanwhile, whereas the ‘process’ of development is very important with custom-made software and its implementation, in market-driven development of software, the consumer cares more about the product, and less about the process taken to create the product.

2.4. Packaged Software Challenges

In the second of two studies, Xu & Brinkkemper (2007) identify some challenges involved with packaged software development such as requirements, delivery, and implementation. Packaged software firms face a major pressure, which relates to 'time-to-market'. This can affect the product strategy, because external pressures may mean that only a limited number of requirements can initially be implemented before the product release. Therefore, it is essential to create an order of priority when it comes to customer requirements (Xu & Brinkkemper, 2007; Karlsson et al., 2003). In short, it is important to determine when packaged software will be released, the features the product will have, the associated development costs, and the resulting product quality. Packaged software is often offered to a market through cumulative releases, with each release involving a significant increase in functionality. This requires careful release planning and requirements prioritisation, so prioritising such requirements is an important process in developing the business strategy of a software firm. Release management is therefore concerned with the management of many tasks, including planning, building, testing, version control, and configuration management (Xu & Brinkkemper, 2007; Karlsson et al., 2007).

It is usually too expensive to customise software to suit every potential customer, so packaged software is often delivered to a number of unknown customers and implemented on large numbers of unknown systems. Factors relating to delivery to market are consequently of great importance to the development of packaged software (Xu & Brinkkemper, 2007).

Chaisson & Green (2007) also conduct research in this area. They begin their study with the question: "Can I use an available off-the-shelf package to do my task?" They note that when asking such a question, one must already make some assumptions about requirements-driven design, and be ready to reconsider assumptions. Organisations will inevitably have specific needs and requirements that must be met by packaged software. These will either have to be met via the product's existing functionality or by customising the software (Sia & Soh, 2007). As Chaisson & Green (2007) note, the discrepancies between the software functionality and the practices of the organisation using it are generally a matter of the software being "too far" or "too close". The software can be "too far" from the specific needs of the organisation and therefore require extensive reconfiguration. Or, the software functionality can be "too close", because of irrelevant or inappropriate functionality, and sometimes this functionality will not be able to be modified. Chaisson & Green (2007) conclude that due to the broad audience intended for the pre-existing packaged software produced by software companies, ensuring software 'fit' may require an organisation to change their own organisational practices, just as much as, or rather than, modifying the software to fit the organisation. Therefore, requirements may actually be determined by what the software can do, more so than by what the organisation does or wants to do.

2.5. Packaged Software & Requirement Engineering

It has been established that Requirement Engineering (RE) for packaged software is different than for bespoke software projects (Karlsson et al., 2003). Time-to-market and the identification of an insufficient selection of initial requirements are two major challenges faced by packaged software projects (Sawyer, 2000).

Several studies have investigated the processes related to packaged software RE. For example, Mishra & Mishra (2008) suggested that use should be made of the dynamic requirement engineering approach of agile methods when developing packaged software. This approach can be used to handle the challenges of time-to-market and insufficient initial requirements. They conclude that software should be released in increments, with higher priority requirements being met in earlier releases. Low priority requirements would be excluded from early releases, but implemented in later releases.

Meanwhile, feedback from the earlier releases can help with refining the original requirements that were met, and with recognising or adding new ones.

Another study, conducted by Regnell et al. (2001) investigated the packaged software requirements engineering process. Regnell et al. (2001) point out differences between bespoke RE and packaged software RE, stating that in packaged software RE, requirement gathering occurs with the input of a defined set of users or in relation to an imagined group of people who may fit the profile of an intended product user. Often, requirements are invented by developers, based on strategic business objectives, domain knowledge and a product vision. This means that in order to create packaged software with value, companies need to place further emphasis on the identification and selection of requirements before developing projects.

2.5.1. Packaged Software Requirements Engineering

One study by Daneva (2004) that explores what elements constitute the RE process for the implementation of ERP packages leads to the suggestion that “we know little about the issues arising when organisations make the standard RE model a life process, and we know less about how to make such a process work better”. The information Daneva (2004) supplies is based on 13 different implementations of SAP carried out by herself and a project team at Telus Mobility during 1997-2002. The project team followed guidelines from the ‘Requirements Engineering Good Practice Guide’ in order to assess various situations encountered. From these experiences, the team was able to assess which RE practices worked, which ones did not work, and how to improve the generic process model of RE.

The information learned by the team through these experiences included lessons related to organisational issues, infrastructure, re-use, and turning a ‘standard model’ into a ‘live process’. With regard to organisational aspects of ERP process support, these lessons included reducing barriers to cooperation, creating win-win partnerships, and streamlining knowledge transfer from consultants to clients. In terms of process infrastructure, Daneva (2004) recommends that organisations “use the vendor’s architecture framework" because basing blueprint versions on the vendor's framework will help an organisation to manage requirements complexity, will provide them with common terminology, and will allow them to meet at least the basic requirements of the business. After the basic requirements are met in an initial blueprint, the existing architecture can be used as a base from which to incorporate "more sophisticated requirements" or to "address controversial ones". Daneva (2004) also states that RE teams need to understand RE process-tool dependencies and RE standards use. If these "tools and standards are new to the client's organization", they should be introduced "in parallel with the RE process". She suggests that it may be helpful to "design scenarios of how to use tools and standards to support specific RE activities".

Daneva (2004) also considers "requirements reuse aspects" of the ERP RE process, including the importance of applying a "reuse measurement instrument". She suggests that when a reuse measurement process is established as part of the RE process, this helps an RE team "adopt or adapt standard reuse counting practices in ERP RE". Daneva (2004) and her team found that by using a reuse measurement instrument they could improve the quality "of five RE deliverables", which included "business blueprints, business process models, data models, project plans, and project estimates". Daneva states that reuse risks must be assessed early on during an ERP implementation. Not all data and process components can be reused and problems can occur if "fully reused, customized, and newly created requirements" are mixed together without much thought or if possible reuse options are ignored. Daneva (2004) suggests it is necessary to consider the "costs and residual risk of each possible reuse-handling option" and that "when process owners better understand ERP reuse and customisation risks, they're less inclined toward unnecessary adaptation and will willingly reprioritize requirements". According to Daneva, "reusing process and data requirements requires the client's commitment to the ERP package's default processes, integrated data flows, and data-sharing mechanisms". Without this commitment, packages might be subjected to "extra-ordinary customisation", thus defeating much of the purpose of adopting an integrated ERP system. Issues such as business process and data standardisation should be determined early on during requirements elicitation meetings.

Daneva (2004) also discusses process aspects of RE: the practices an RE team should put in place to "support the key activities of requirements elicitation, documentation, and negotiation". She observes that it is advisable to "systematically validate and verify requirements", in order to avoid skipping over validation efforts or not considering potential clashes of functionality or of business drivers. She defines "requirements validation" as "the process that ensures that business requirements clearly describe the target solution", and "requirements verification" as "the process that confirms that the requirements are technically implementable and that the resulting architecture design satisfies the business requirements". One way of assisting this is by organising "structured process validation walkthroughs" to explain how the system is supposed to work. Another useful practice is to document the rationale for requirements. It may also be useful to involve a "data architect" in the RE processes, asking them to create entity-relationship data models, conversion plans, and interface specifications. Daneva (2004) also considers it important to hire an ERP consultant with experience in requirements modelling with the ERP package in question and with "designing business processes in the client's cultural context". If this is not done, any external consultants hired may exchange in only very rudimentary modelling, limiting the benefits the organisation can receive from "RE documentation methods and tools". Meanwhile, in order to prevent a "creeping" level of requirements or requirements coming from unofficial sources, RE teams should use "existing architecture artifacts, RE tools, and standards".

Daneva (2004) also speaks of the need for "change impact assessment". Impact analysis should be carried out before suggested changes are made. In order to deal with changes, the RE team should focus on "what process owners need to accomplish their daily tasks, why they need it, how they use information, what problems they experience with the existing system, and what must be changed", and on whether a specific change will affect the integration between various ERP modules. Such considerations will allow consultants "to estimate configuration efforts, costs, and customisation risks". Daneva (2004) concludes that if RE practitioners adopt the suggestions and solutions offered in her study, they are "likely to generate more effective and mature RE processes […] when using a generic off-the-shelf RE model". However, Daneva's (2004) study was of large organisations and therefore, there is still a need to understand the approach within small – medium sized organisations. Moreover, her study was conducted at the client/users' site, whereas it may be helpful to look more closely at the perspective of the software company.

In another study, Daneva (2007) argues that those organisations implementing ERP packages are implicitly "adopting standard ERP-vendor-specific process models for engineering their requirements". She notes that it can be very difficult to create a live process out of such models and suggests various ways in which organisations can create more mature models of requirements engineering.

Daneva (2007) notes that "a good practice-driven RE model" is "assumed to ensure that the ERP adopter gets more predictable process results and increased chances of process success". She notes that while it has been shown from past results that "instantiating a standard RE model" may bring benefits to an organisation, it is also very difficult to actually adopt a standard ERP RE model. Daneva (2007) therefore identifies a gap in the literature and knowledge surrounding ERP implementation, as hardly any previous studies actually focus on "the mechanics of the ERP RE process itself and how it leads to successful outcomes". The information used in Daneva (2007) draws on eight years of experience carrying out ERP RE in a Canadian company, assisting with SAP R/3 implementations. Daneva's 2007 study therefore aims to answer the questions of "(1) what do more successful and less successful RE process instances look like? (2) How does degree of success or failure vary? (3) Is the distinction between RE success and failure immediately evident, and if it is, then to what extent?"

The ERP implementations considered by Daneva (2007) used "the Accelerated SAP (ASAP) RE process, engineered and standardized by SAP". The case studies represented by Daneva in this article largely repeat information from Daneva (2004), except that another 87 subprojects and "process instances" within the telecommunications company (Telus) were considered. She focuses particularly on the 56 process instances that were cross-organisational (being implemented in "at least two departments or business units" of the organisation).

Daneva (2007) notes that in the literature surrounding RE, RE process is considered successful if the following goals are achieved: "(1) business requirements are defined and on-time and within budget; (2) expected architecture design is delivered; (3) project resources are used efficiently; and (4) RE process stakeholders are happy".

In order to understand what happened during the ASAP live process, "we systematically assessed the process instances that we observed in each project by means of a standard RE maturity framework, namely the Requirements Engineering Good Practice Guide". This framework allowed Daneva to assess "maturity" levels of how an organisation uses and follows RE practices, to also investigate which RE practices tended to be skipped, and to consider "completeness and consistency aspects of the requirements documents". Using "both quantitative and qualitative" information, Daneva (2007) characterises "common points of success and failure".

Daneva (2007) explains that it was considered that RE process success occurs when "overlapping goals were achieved: timely and cost-effective delivery of requirements, correct architecture design, happy stakeholders", and that an RE process failure was deemed to have occurred when there was a situation in which "there was a combination of: missed deadlines, budget overruns, decreased consistency and completeness of the requirements, increased stakeholders' dissatisfaction, and rework in the later project stages due to poor requirements". Daneva (2007) found that instances of "catastrophic failure" during an RE process were infrequent, and that those which occurred were usually caused by "unnecessary implementation of complex functionality", requirements which "overlooked critical architecture design", requirements which lead to "massive customisation", instances where change impacts were underestimated, and instances where requirements could not be implemented because of the "built-in assumptions of the package". There were also situations where there was one visible cause for the failure, such as when process modelling activities were skipped or when possibilities to reuse requirements were ignored in favour of customisation.

Daneva's (2007) study considers how such instances of success and failure mapped onto "the REGPG [Requirements Engineering Good Practice Guide] process maturity levels". They found that "merely bringing in the generic process model is not enough" and that those teams that focused heavily on requirements elicitation were also those whose processes had visible and catastrophic failure. These groups spent "relatively little time" on documentation, modelling, and negotiation activities. The groups with visible and resounding successes were those that placed a great deal of effort into negotiation activities rather than those that expended the majority of effort carrying out elicitation tasks.

In conclusion, Daneva (2007) delivers both predictable and surprising findings. Some of the more predictable findings are that protracted requirements elicitation has an association with process failure, and that context factors such as a project team's awareness of RE processes and their experience with business integration can affect process success. However, some results were more surprising, such as that "most RE processes that didn't prioritize cross-organizational ERP requirements were successful", and that a process could have a resounding success even in the absence of "a requirements change impact analysis" having been carried out.

Meanwhile, Niu et al. (2011) focuses on how an organisation can exploit their chosen ERP Requirements Engineering model to frame how they engage in business application development. The exploratory case study discussed by Niu et al. (2011) shows how "aligning business applications to the packaged RE model" can lead to "integral practices and economic development". Niu et al.'s case study examines what happened within one large IT company when they followed the RE practices recommended by Oracle's AIM method. The aim of their paper is to encourage RE researchers toward making empirical studies of ERP RE adoption and to develop "more effective and mature processes when exploiting ERP RE methods". They argue that "the effective use of ERP software requires a systematic method that facilitates the exploitation of the benefits of ERP functionality while guarding against the technical and business pitfalls. This method should ideally be applied during requirements engineering (RE) activities".

Niu et al. (2011) provide a figure which demonstrates mismatches that can occur between a client’s requirements and wishes regarding a system and what is actually provided by the pre-packaged system. It demonstrates that even when customisations are made, the client organisation may still end up with unsupported needs, or that a customisation may provide functionality that is not needed. In their study, they argue that choosing an ERP system with the best ‘fit’ for an organisation, or tailoring it so that it has the best fit, involves choosing a system that allows the organisation to maximise reuse of the provided functionalities and then tailoring business application development so the organisation’s business processes begin to align with the functionalities of the package. Niu et al. (2007) state that literature on RE has spent very little attention on the second criterion mentioned here: “how to leverage the selected ERP package to construct high quality business applications”. In their study, they suggest that one way to align business requirements with the ERP system is “to take advantage of the RE guidelines defined in the chosen ERP package”. However, the study by Niu et al. (2011) does not consider any aspect of choosing the ERP system but rather studies how the ERP applications were developed as the company used the Oracle application implementation method (AIM).

Niu et al. note that one way to take full advantage of a chosen ERP package “is to adopt the RE guidelines defined in the ERP package”. Most leading ERP packages feature such pre-packaged methods, and following these methods is more suitable than following RE processes designed for bespoke software. Niu et al. (2011) state, however, that this does not mean that customisations cannot be made.

Niu et al. (2011) carry out an “exploratory case study” which involves analysing the ERP implementations carried out by “ZT”, a PC manufacturer and IT service provider headquartered in Beijing. They note that their exploratory study reveals some of the implicit assumptions of the Oracle application implementation method (AIM) that need to be made more explicit. For example, ZT found that they needed to add further artefacts and processes to better support data visibility and integrity. They also found deviations “from ERP RE practices in the literature”: for example, ZT found that it would have been better for them to have had a complete set of requirements before starting their projects – this suggestion conflicts with some previous literature on ERP RE. Niu et al. believe that, in general terms, following the RE processes recommended by an ERP system (such as Oracle's AIM methodology) will help with requirements alignment and lead to integral RE practices and to economic development. They believe that it is best to stick with the ERP vendor's RE approach when creating the first blueprint of a system and that "more sophisticated requirements could be gradually incorporated" later on. It should be noted, however, that Niu et al. (2011) consider three ERP implementations that were carried out according to accelerated methods and that involved implementing only limited functionality. Their study also focuses on a "relatively immature organisation".

2.5.2. Packaged Software Requirements Prioritisation

According to Wiegers (1999), requirements prioritisation is needed to indicate how essential each requirement, feature, or use case is to a particular product release. If all requirements are considered equally important, it will be difficult for the project manager to respond to budget cuts, schedule overruns, personnel losses, or any new requirements added during development. The prioritisation of requirements before projects and allocating them to releases is an essential activity that provides value for customers (Lehtola & Kauppinen, 2006). Wiegers (1999) also suggests that priorities should be adjusted periodically throughout the period of development as customer needs, market conditions, and business goals evolve.

Barney et al. (2008) stated that the purpose of requirements engineering activities "is to add business value that is accounted for in terms of return on investment of a software product". They present three case studies of release planning processes for packaged software and find that the client and market base of the software product represents the most influential group in the decision to implement specific requirements. This is reflected both in terms of deciding the processes followed and the criteria applied when selecting requirements for the product or for a specific release. However, the maturity of the product, its marketplace, and the development tools and methods available also influence whether or not a requirement is included in a specific project or release.

In a very recent study, Rowland (2012) considers the process of "fit-gap" work often carried out by committees set up to decide how to prioritise customisations and work-arounds that may be needed during ERP implementations. Rowland's study borrows the terms "fitting" work from Gasser (1986) and "fit-gap" from Rowland & Gieryn (2008) and "examines the process and consequence of prioritising implementation projects with special emphasis on 'de-prioritisation' as a practical technique for managing installation". In order to make packaged software work for their company, companies usually engage in a mixture of formal "fit-gap procedures" and off-the-cuff work-arounds. Project managers of such implementations often create "fit-gap committees" that involve functional and technical employees. Sometimes the work-arounds and customisations carried out are needed in order to "coax suboptimal implementations into functioning properly". However, Rowland identifies a problem related to the issue of de-prioritising the need to fix some of the gaps that are identified during implementation. De-prioritisation of requirements is often used in order to limit the number of customisations made. But Rowland (2012) states that problems can arise when time and resources that could be invested into customisations are delayed indefinitely or when potentially helpful work-arounds are not carried out. Rowland suggests that many such decisions to prioritise or de-prioritise are actually caused by "political conflict and ambiguous economic accounting". Rowland's investigation of a "multiyear organisational case-study of ERP in an institution of higher education" reveals the process of how those involved in the ERP committee prioritised or de-prioritised some "gaps".

Rowland & Gieryn (2008) explain that fit-gap work is "a formalized process devoted to (1) identifying gaps, (2) deliberating on possible fits, and then (3) prioritizing which gaps are fit straightaway while other get de-prioritized". Rowland argues that conceptually, the fit-gap idea seems valuable. However, fit-gap committees cannot guarantee that they will discover "all gaps in fit", and some work-arounds may be only be “patch-work” that might “shore-up lingering misfits” (Gasser, 1986; Pollock, 2005) or work only until the next vendor update.

Rowland next suggests that while past research on modifying ERP systems has focused on “notions of fit and alignment”, it could be fruitful to instead think in terms of ‘gaps’ and about competing claims regarding “what constitutes a legitimate gap”, which gaps must be closed, and which are safe to leave open. He suggests that customisations and changing work practices can both be described as “fitting work”, to use a term created by Gasser (1986). He states that this fit-gap work is “emergent, recursive, and ongoing”– even after every module of a new system is installed.

Light (2005) also points toward the more political nature of how misalignments may be dealt with. Light finds that “managers and consultants often frame misalignments to be economic, functional, or technical in nature”, but that these descriptions work to distract from the politics that may be involved with decisions about misalignments.

Rowland & Gieryn (2008) observe that gaps might appear when universities or colleges implement ERP systems because even though institutions of higher learning would obtain necessary information about the institution’s extant practices and what was needed from a system before implementing a new system, problems necessarily arise because employees share “tacit knowledge” and “embodied practice” and it may be difficult to “retrieve and articulate” this knowledge precisely. The institution of learning can therefore only consider “a truncated subset of all the possible differences between the legacy system and [the packaged software solution]” (Rowland & Gieryn, 2008, p. 384). Meanwhile, a study by Kitto & Higgins (2010) found that work-arounds that helped universities keep legacy processes were not always liberating.

Rowland (2012) argues that there are sometimes contradictory or multiple rationales behind efforts to prioritise gaps for fit. He ends his study by noting that once ‘gaps’ are identified within organisations, various ‘fits’ are identified for those gaps, with only one of these ‘fits’ being prioritised within a queue. It is also usually the case that the number of fits that could be “immediately available for implementation” will involve a cost in excess of the amount of financial resources available. Rowland therefore argues that in such cases, “a valuable strategy for managing implementation projects is to de-prioritize some fits”. However, the political or economic scheming that might be involved in such decisions is generally hidden and such decisions made to appear neutral. Rowland argues that now he has emphasised how ‘gaps’ may be constructed and dealt with by implementation committees, future research into ERP implementation should not focus only on ‘fit’ and alignment, but should also “recognize that system fitness and failure are also predicated on how gaps are locally constructed and prioritized”.

Underlying the importance of requirements prioritisation in packaged software is the fact that there are limited product development resources. Time and money are both finite, and when customer expectations are high, timelines short, and resources limited, the most essential functionalities of the product should be delivered as early as possible (Wiegers, 1999). The product should, however, not only reach the markets in time, but also meet the needs of the customers (Karlsson et al., 2007). Therefore, a balance must be struck during the development of the product, with the most important requirements being found. The question here is “how do software companies balance between packaged software functionalities and meeting clients’ requirements?” In order to manage the balance between software specificity and generality successfully, domain knowledge and estimation skills are required.

2.5.3. Requirements Engineering in SPSVs

Aranda et al. (2007) investigate how small companies elicit and track requirements. Their study presents “preliminary results from an ongoing exploratory case study of requirements management in seven small companies” and is the first study to look specifically at how small companies “elicit, track, and communicate their requirements” and at their “contexts of operation”. Aranda et al. find a diversity of approaches to requirements engineering within these companies. Despite the fact that these small companies seemed to lack any systematic approach to eliciting and tracking requirements, the companies do “well enough” at this stage of RE and any errors in requirements are “rarely catastrophic”.

Aranda et al. (2007) initially based their study on the hypotheses that requirements engineering in small companies did not appear to follow any best practices laid out in textbooks and that the current research on requirements modelling, specifications, and traceability "seemed to be irrelevant to these companies". In their exploratory case studies of the seven companies selected, they aim to answer three questions: "How do small companies manage their requirements?", "How does the context of these companies affect them?", and "Why do these companies adopt some requirements practices and reject others?"

Aranda et al. (2007) explain that each company studied had developed their own practices that worked for them: the practices allowed them to stay afloat, to communicate with each other, to create what customers wanted, and sometimes to have business growth. There were very few common features of the requirements processes practiced throughout the seven companies. Each addressed "the issues of elicitation and communication of requirements with different degrees of planning, structure, and documentation"; they also approached the mitigation of errors in different ways, such as discovering them through the iteration process, through the use of demos, through beta testing, or through "upfront analysis". Some variables did seem to affect the companies' choices about how they carried out requirements engineering. Business and contextual factors such as "the type of customers, the background and skills of their developers" and even "the spatial layout" of their offices seemed to influence decisions about requirements engineering, but Aranda et al. (2007) consider that they do not yet have enough evidence about the influences of each factor to state anything certain about them. They do, however, hypothesise that "the diversity of RE practices in small companies can be explained as the result of evolutionary adaptation, as these companies have adapted to a specific niche". This is important, because if correct, then "no generalized requirements technique will be suitable for all small companies".

Each one of the seven companies had a story to tell of how errors in requirements caused problems with a project. But in no case did the requirement errors ever lead to a catastrophe. Aranda et al. (2007) deliver three hypotheses in relation to their findings:

first, that “small companies that survive their initial phase practice normal design, which greatly decreases the risks associated with software engineering”; secondly, that “small companies can fix their requirements problems more easily than large companies by virtue of being small”; and, thirdly, “a single requirements catastrophe will drive a small company out of business”. In relation to the first hypothesis, Aranda et al. (2007) state that the adaptation practiced by small software companies may include “a shift from a radical design to a normal design approach to software development”. In relation to the second hypothesis, Aranda et al. (2007) suggest that it may be easier for small companies to clear up requirements errors or misunderstandings due to the smaller scale of many of their projects, the greater ease of arranging meetings about requirements, and the use of open office plans. Lastly, even when problems arise with requirements, small companies choose to maintain the same requirements processes they have used in the past. Aranda et al. (2007) believe that the governing principle behind the requirements practices of these small companies is to find a process that is “good enough” and that they will resist any radical change to their requirements techniques.

Aranda et al. (2007) conclude that small software companies have a number of special characteristics that distinguish them and their requirements processes from those of larger companies. Aranda et al. (2007) therefore propose that when writing about RE and recommending specific RE practices, researchers should “state the context” in which it is believed the techniques would be helpful, that researchers should “connect RE research to business and social concerns” of small companies, that researchers need to provide compelling reasons why a small company should abandon their current RE techniques in favour of techniques proposed by the researcher, and that it may be better to propose incremental changes to the current set of practices, rather than full-scale change which could have negative effects on a small company.

Bürsner & Merten (2010) observe that SMEs may not always be able to apply RE methods and techniques designed for larger companies without having to modify them. They also point out that SMEs face particular difficulties caused by time constraints and limited resources. Their study argues that SMEs [SPSVs] need improvements to be made in the methods for "requirements elicitation, documentation, communication and traceability" and in requirements management throughout the software lifecycle. In their study they discuss a workshop (1st Workshop on Requirements Engineering in Small Companies (2010)) which was held to investigate "the RE practices and experiences of SMEs [SPSVs]", and to discover or discuss "lightweight RE methodologies and tool support usable for small companies".

During the workshop it was found that "existing RE techniques are not sufficient for small companies". Other recommendations made after the workshop are that size is "not the only measure to categorize smaller companies and describe the exact focus of research", "that tacit knowledge and social structures" in place in SPSVs may play an important role in RE research, and that introducing RE methods designed for larger companies may actually do harm to the specific features of an SPSVS.

In the same year, Quispe et al. (2010) noted that previous researchers had identified "a lack of knowledge about requirements engineering practices" in small and very small software companies. Quispe et al.'s paper presents the results of a diagnostic study relating to very small software companies (VSSE) in Chile. The results and discussion presented draw on data gained through various focus groups and surveys. Their study focuses on the current state of RE practice in such companies, identifies "common areas of improvement", and considers the limitations that might arise when trying to "adopt appropriate requirements engineering practices" in very small software enterprises. While Quispe et al. (2010) accept that various factors could contribute to the failure of projects within these companies, they choose to investigate the companies' requirements engineering practices since RE is known to affect the outcome of projects. Quispe et al. (2010) delineate some of the problems that can be caused by inadequate RE practices. These include having to "rework" parts of a system as new requirements become apparent late in the project, problems with coordination and communication, and problems with the visibility of the project status. They point out how "reworking" can significantly delay a project. Coordination and communication problems may occur in the way that the VSSE manages various documents, slides, emails, and other resources relating to requirements. There may be difficulties with obtaining on-time, fast, or accurate information on requirements. Some projects carried out by VSSEs may lack the requirements-related metrics that could help to steer a project toward successful completion.

However, Quispe et al. (2010) conclude that requirements engineering practices can only be improved if investigations successfully identify areas for improvement, and that it is very unlikely that a one-sise fits all approach is a suitable choice for very small software companies: it is more likely that there will be a particular answer for each individual company. Their study is among the first to investigate RE practices within VSSEs, preceded only by a study by Aranda and Easterbrook in 2007. Quispe et al. (2010) briefly discuss Aranda and Easterbrook's results, noting that they reflect investigations into only 7 different companies. Apart from the study by Aranda and Easterbrook, they were unable to discover any extensive studies of RE practice in VSSEs; they did, however, find some studies relating to how VSSEs conducted RE in specific scenarios.

Quispe et al. do mention seven specific findings from their focus group: during projects carried out by VVSEs, (1) project specifications were usually met, but the client often found the solution unsatisfactory; (2) since solutions were often unsatisfactory, it is likely that there was inadequate communication with the client, leading to incomplete specifications; (3) the project's scope often expanded, as clients asked for additional changes – meanwhile, these changes were often inadequate; (4) VSSEs often carry out an ad-hoc form of requirements specification; (5) ad-hoc processes often lead to a loss of requirements and other requirements management issues; (6) developers often resolved issues of uncertainty without informing the clients; (7) and that "VSSEs are aware of the benefits of RE practices but are not sure they apply in their context" (83).

A study by Aranda (2010) argues that researchers should stop encouraging small companies to follow RE practices that were designed for larger companies. It is argued that instead, small companies should take advantage of the special opportunities they have "to develop software efficiently and successfully". Small companies may have particular strengths, such as the ways that they engage in “requirements elicitation and communication activities”, and researchers should respond to such strengths by designing or evaluating techniques that make the best use of a company’s small size.

While arguing that small companies should take advantage of their strengths, Aranda also asks “when does a growing organization cease to be small? That is, when do these strengths disappear?” He observes that there often seem to be changes in company dynamics when a company reaches ten to twenty people, and again when it reaches about one hundred and fifty people”. Future research, therefore, could explore these thresh-holds. Aranda (2010) also states that perhaps it is not only the number of employees that should be considered in studies of small companies, but also “other determinants of size, such as the number of teams or the number and variety of customers”.

Merten et al. (2011) identify a lack in the amount of research conducted that helps to categorise SPSVs. They also identify problems related to the way that SPSVs are categorised and the way that the results of observational studies, field studies, and empirical studies relating to SPSVs are classified. They point out that a “single criteria cannot be used to define SMEs [SPSVs]”; rather, additional criteria that influence RE practice should be identified and classified, in order to better be related to SPSVs. Their study brings up the need to provide initial attributes for identifying and classifying SPSVs and ways of improving RE techniques within the companies. Their study does not however deliver any final results from such research, but rather delineates the issues that their intended study will deal with. At the end of their study they indicate their plans to disseminate a questionnaire amongst a large range of SPSVs in order to “correlate sets of parameter values within the attributes with RE practices used in the companies” and then formulate hypotheses from the results. The research in their study is intended to be specifically directed toward studying SPSVs.

Merten et al. (2011) suggest that software engineering SPSVs are generally rather flexible, agile, and innovative, and well-known for “advanced software engineering competence". However, as they mention, the field of SPSVs and how they should practice software engineering is still under-researched.

In their discussion of the problems surrounding studies of RE in SPSVs, Merten et al. (2011) state that a definition for SMEs that relies on the number of employees is insufficient for categorising SPSVs. Problems related to this insufficiency result in studies being classified as relating to SPSVs whether the research involves companies that have only 4 employees or more than 150 people. Therefore, previous findings often "lack a reliable set of criteria identifying the kind of SME they are referring to", and the results provided may seem conflicting. As a result, it is often unclear exactly which kinds of SMEs various process models or best practices are intended for. Moreover, past studies do not seem to consider what different kinds of software might be developed by these SPSVs and how this might affect their ability to implement different RE techniques.

As mentioned, Merten et al. (2011) do not deliver any final results from investigations into these issues, but instead delineate the questions that they plan to investigate during their study. For example, they discuss their intention to discover whether or not developers at SMEs are right to consider that soft factors such as their domain knowledge or the particular niche that their SPSVS deals with limits the need for formal methods of RE practice and management. They also discuss their intention to formulate a questionnaire once new attributes of various kinds of SPSVs have been identified, and send the questionnaire to a large number of SMEs in order to gain further information that will allow them to map SPSVs' RE practices with their various attributes.

2.5.4. Packaged Software Requirements Engineering in SPSVs

In a study functioning as part of the 'Third Generation ERP Project' that aims to develop a standardised and more cost effective ERP system for SMEs, Johansson & Bjorn-Andersen (2007) consider the challenges that are involved in identifying "business requirements for a future standard enterprise resource planning (ERP) package" that can support the needs of SMEs. They argue that the success of such a complex project likely relies on “the success of the requirements modelling”. Their study therefore examines various models for identifying “needed and future business requirements”.

Johansson & Bjorn-Andersen (2007) explain that in the past, ERP vendors have usually pitched ERP systems to SMEs by first supplying them with small versions of ERP based only on accounting systems, and then sometimes gradually extending the system by adding functions related to supply chain, logistics, and CRM. Even so, the implementation costs for an SME using ERP are quite high. Therefore, as Johannson & Bjorn-Andersen explain, the objective of the Third Generation ERP Project (3gERP), is to develop a “comprehensive global ERP system” that could be used by companies within any country, after only a limited number of modifications and on-going efforts. One key to project success is the ability to identify what elements are needed in an ERP system for SMEs (as distinct from existing ERP systems). Some aims for the new project are to develop an ERP system that is easier to implement, that “may be distributed globally at relatively low costs”, that will be easy to localise, maintain, and update, that will allow enterprises to collaborate, and that “will provide better business insight (data mining) for managing the enterprises”. Two important factors for developing such a system are the ability to successfully identify the business requirements of SMEs, and the ability to present those requirements: two tasks that Johansson & Bjorn-Andersen admit might prove impossible.

There are other key problems involved with developing a standard ERP system. One is that ERP systems are typically developed by a vendor without any interaction with end users. Secondly, various stakeholders involved with an ERP system may disagree on the importance of various requirements. There are, for example, often ‘gaps’ between what business analysts think a company needs and what the management of a company wants from its ERP system. Therefore, one of the questions addressed by Johansson and Bjorn-Andersen is whether there is a way to “develop some kind of process for requirements identification, collection, and presentation” that will future ERP systems provide functionality that is closer to what end-users want.

Johansson & Bjorn-Andersen (2007) suggest that the categories of misfits identified by Soh et al. (2000) may be helpful when seeking the requirements of a future ERP system. Soh et al. (2000) placed misfits into three categories: "data, process, and output". Data misfits relate to the data format and underlying data model of a system. Process misfits often relate to issues with licences and access to various functions or to inability to change the source code. Operational misfits occur when the ERP system does not support a particular operation that is normal for the company. Output misfits are described by Soh et al. (2000) as occurring when the company does not receive the information it needed about the ERP system. Soh et al. use the common definition of 'misfit' as a gap between the functionality offered by a software package and the functionality required by the company.

Johansson & Bjorn-Andersen (2007) note that there is another way to analyse requirements: by considering the company's "business model as well as the nature of the businesses. Requirements could be thought of as intra-organisational or inter-organisational. Requirements will also be different depending on whether the organisation is a government organisation, service organisation, or manufacturing organisation", for example.

Johansson & Bjorn-Andersen (2007) present their own model for identifying and presenting business requirements for ERP systems. Their model builds on constructing narratives about an ERP system and its use. Much information about requirements can be gained from the end users (via interview) and from the existing ERP system, even if it is hardly used. Scenarios about the current ERP system can be developed into scenarios about a future system, especially once input is obtained from ERP-related literature or experiences of previous implementations. The scenarios that are developed should then be related back to the company's goals and what is needed to fulfil those goals. The process therefore involves "reverse engineering from the scenarios".

As observed by Johansson & Bjorn-Andersen (2007), any future global ERP system will need to be able to deal with local issues such as tax laws, different accounting standards, local practices, and various languages. They also mention a world-wide trend towards virtual business environments, and other forms of technology that might affect the development of a new ERP system, such as "enterprise application integration (EAI), extended markup language (XML), service oriented architecture (SOA)" and 'software as a service (SaaS)".

Another study by Vilpola et al. (2007) discusses the implementation of ERP systems in SMEs. Noting the need for SMEs to use an ERP system that is able to meet the requirements of their processes and allow them to keep their individuality, Vilpola et al. (2007) develop a Customer-Centered ERP Implementation (C-CEI) method. Their C-CEI method involves three stages of analysis, "operational, contextual and risk", that they believe will help an SME select an appropriate ERP system and better ensure the system's acceptance by users within the company.

Vilpola et al. (2007) stress how implementing an ERP system almost always means a compromise between a company's existing processes and what the ERP system can do. They state, therefore, that "in order to select an ERP system that best supports the business processes of an SME, an in-depth analysis of company's processes and requirements is needed. The analysis must support both the ERP system implementation process planning and process change management". The implementation method developed by Vilpola et al. innovatively combines "two requirement engineering approaches". The first is a focus on "company operations and processes", and the second a focus on "users and their tasks in the context of use".

The C-CEI method involves three major phases of analysis: operational analysis, contextual analysis, and risk analysis. After each phase, a document is created which can be used to assist implementation. Vilpola et al. (2007) argue that this method best prepares the staff of an organisation for an ERP implementation as they are involved in each phase, including describing current processes and problems, modelling the context of the system, and prioritising the implementation risks. Vilpola et al. (2007) state that the C-CEI method is a multi-disciplinary approach to ERP implementation, and may require small and medium-sized companies to hire consultants. However, the method should result in such companies obtaining very thorough information about ERP requirements and about the processes and environment within their company. Vilpola et al. emphasise that the C-CEI method is innovative in its focus on user-centeredness, and they note that the C-CEI method could be further developed through the creation of a tool-box containing instructions, checklists, and templates. Lastly, Vilpola et al. (2007) note that the C-CEI method can reveal ways for improving a company's processes that are separate from improvements that rely on implementing a new system. The C-CEI method can even supply information that may help a company after the implementation of a new ERP system.

More recently, Jantunen (2010) argues that small and medium-sized companies require a form of RE that allows for the flexibility and sociability that are frequently features of SMEs, which allow for improvisation or that help employees "cope with multiple meanings". It suggests that the management of requirements in small and medium-sized companies depends to a large degree on human collaboration. However, when companies grow they often lose some opportunities for face-to-face collaboration. Jantunen (2010) therefore explores how companies keep "the benefits of human collaboration while coping with increased complexity".

The first topic covered in Jantunen's study is Market-Driven requirements engineering (MDRE). When companies plan to offer a new software product, they must determine which new features should be included. MDRE processes can be described as "approaches to synchronise" "candidate requirements" "with the discrete release events". However, past studies have noted that determining the best possible selection of features for future releases is extremely difficult, and that there is no optimal solution. The criteria that could be used to determine the success of the solution are always changing, as technologies, market needs, and competition change. Moreover, if new customers are acquired, their demands may conflict with the demands of other customers.

Jantunen (2010) discusses the process-based MDRE approach, noting that companies receive information about market demands from many sources: their current customers, trade shows, and from watching what competitors are doing. Many companies place the market information they have collected into a database. However, they often collect more market information than they are able to digest. Meanwhile, it is difficult to know which customer requests to prioritise and which would most benefit the company. The process of deciding which requirements to use tended to involve scanning the database for requirements. A “release plan” is created showing the features to be included in a particular product release, and a “roadmap” created showing “a layout of the product releases to come over a time frame of three to five years”.

Jantunen’s (2010) study concludes that “as long as an organization is small enough, human collaboration appears to have a natural tendency to mitigate the MDRE-related challenges. When the organisation grows, it begins to face coordination challenges”. In order to meet those challenges, the organisation often turns to “introducing processes” and increasing documentation. The paradox is that this attempt to increase coordination can lead organisations to lose the benefits of human collaboration. Without such collaboration, employees may end up working “with partial information”. Since “organizations need to find new ways to regain the benefits of human collaboration while coping with increased complexity”, current MDRE approaches could be expanded to include approaches that allow for multiple meanings, that are tolerant of improvisation, or that are “more social by nature”. Social media may assist “social interaction, content sharing, virtual identity and collaborative production”. Jantunen suggests that further attention should be paid to studying small companies and that researchers should explore how social media could help larger companies to retain the benefits of human collaboration.

2.6. Packaged Software Implementation

Implementing Packaged Software (PS) is a major project requiring a significant level of resources, commitment and changes throughout the implementing organisation (Addo-Tenkorang & Helo, 2011). Often the PS implementation project is the single biggest project that an organisation has ever launched, and many cases have been documented of failed implementations leading to the complete business failure of some organisations. As a result, issues surrounding the implementation process have recently been one of the major concerns in the IS industry. A study by Moon (2007) about the literature relating to ERP published between 2000 and 2006 showed that 40% of the 313 articles considered elements relating to PS implementation. Similarly, Addo-Tenkorang & Helo (2011) in their review of the literature relating to ERP published between 2005 and 2010 showed that discussions of PS implementation comprise more than 54% of the entire articles. The challenges related to PS implementation were identified as long ago as Holland & Light (1999), who offer a look at early literature on packaged software implementation.

2.6.1. Critical Success Factors for Packaged Software Implementation

Holland & Light's (1999) study offers a framework for helping managers "successfully plan and implement an ERP project" that will help them with "IT planning and legacy systems management". In their study they focus on the two research questions: (1) "How can ERP systems be implemented successfully?", and (2) "What are the critical success factors for ERP implementation?" Holland & Light identify that the two main options when implementing packaged ERP software are either to install the software as a "standard package with minimum deviation from the standard settings", or to customise the system "to suit local requirements". ERP implementation also involves two main processes: "business process change", and "software configuration to align the software with the business processes" (Akkermans & Helden, 2002; Scott & Vessey, 2002; Umble et al., 2003; Soja, 2006).

The study by Holland & Light suggests that the Critical Success Factor that most influences the success of ERP implementation is "top management support and a clear business vision" (Umble et al., 2003; Soja, 2006). Other important factors to consider when planning for ERP implementation are "legacy systems, ERP strategy, and business process change and software configuration" (Holland & Light, 1999). Holland and Light mention that there are "different approaches to ERP strategy ranging from skeleton implementations to full functionality", but they seem certain that "it appears easier to model the organization to the ERP software rather than vice versa". They suggest that the main questions that leaders of organisations should ask before beginning ERP implementation is whether they have developed detailed plans for the change, whether they have properly considered the effect that the existing legacy systems may have on attempts to change, and whether they should implement a fully-functioning system, or, initially, just a skeleton one.

Somers & Nelson (2001) used the responses given by 86 organisations to a questionnaire about critical success factors in ERP implementations to create a list of the critical success factors they consider most important for ERP implementations. They suggest that the list can help managers of ERP projects to best utilise limited resources by focusing on the CSFs that are most likely to have a major impact on the success of an ERP implementation. Somers & Nelson's paper provides a brief explanation of each of the top 22 factors identified, along with information regarding why the factor was considered important and which previous studies have supported its importance. The top five factors identified by Somers & Nelson are: (1) Top management support, (2) Project champion, (3) User training and education, (4) Management of expectations, and (5) Vendor/customer partnerships. The remaining top 10 factors consist of: (6) Use of vendors' development tools, (7) Careful selection of the appropriate package, (8) Project management, (9) Steering committee, and (10) Use of consultants.

The results provided by Somers & Nelson actually provide two different listings, one showing the top CSFs across the entire implementation process, and another showing which CSFs were rated as most important in relation to the six phases of implementation: initiation, adaptation, acceptance, routinisation and infusion phases. The top 10 success factors for the implementation process overall, in terms of "mean ranking" were identified as: Top management support, Project team competence, Interdepartmental cooperation, Clear goals and objectives, Project management, Interdepartmental communication, Management of expectations, Project champion, Vendor support, and Careful package selection. However, the top CSFs were also identified as changing throughout the process, for example, with "architecture choices, clear goals and objectives, partnership with vendor, and dedicated resources" being identified as the most important factors early on in an implementation. Later on, these factors change. For example, "interdepartmental communication and cooperation" are identified as "important across the adaptation, acceptance, routinisation and infusion phases", even though they had not been considered highly important during the (early) "initiation and adoption phases". "Top management support" was identified as being very important in all but one phase.

Two studies by Akkermans & Helden (2002) and Plant & Willcocks (2007) each attempt to check the accuracy of the list of Critical Success Factors for ERP implementation supplied by Somers & Nelson in their 2001 study. The finding by Akkermans & Helden (2002) is that the list of CSF factors created by Somers & Nelson was helpful and applicable in explaining "both the initial failure and the eventual success of the implementation" within one company in the aviation industry.

The aim of Akkerman & Helden's paper is not simply to test the accuracy of the top 10 list supplied by Somers & Nelson, but to build on the list by theorising some of the "causal interrelations between the individual CSFs". Their two research questions were: (1) "Can the Somers & Nelson list be helpful in arriving at a better understanding of root causes of ERP implementation success and failure?", and (2) if so, "in what way can the Somers & Nelson CSFs be interrelated causally?" Akkerman & van Helden decide to examine the usefulness of only the top 10 ranked CFSs, which, they mention, involve a mixture of 'hard' and 'soft' aspects of implementation. These aspects include such things as "Top management support", "Project team competence", "Interdepartmental co-operation", "clear goals and objectives", and "project management". Some of the items on the top 10 list were initially ranked lower by Somers & Nelson, but moved up the ranking when feedback was received from 52 organisation managers approached by Akkerman & van Helden to provide input. Most of the items on the top 10 list have also tended to appear with frequency in other IT literature surrounding implementation.

At the conclusion of their study, Plant & Willcocks (2007) state that pre-implementation, participants considered three of the critical success factors mentioned by Somers & Nelson to be particularly important. However, by the end of the study, “there was agreement by all of the study participants upon the top four ranked factors as shown by Somers & Nelson’s [list]”. These particular success factors are: Top management support, Project team competence, Interdepartmental co-operation, and Clear goals and objectives.

2.6.2. Packaged Software Implementation ‘Misalignments’

Other studies have discussed packaged software implementation in terms of potential misalignments between the functionality offered by the PS and the business needs of a company. For example, Wei et al. (2005) discuss ERP implementation in terms of potential misalignments between company needs and packaged software functionality, and the resolution of misalignments as the main factor determining whether ERP implementation is successful or not. Wei et al. identify different ‘phases’ of the implementation process and suggest that varying forms of misalignment can occur within each of these stages. The stages identified by Wei et al. (2005) include: the chartering phase, the project phase, the shakedown phase, and onward and upward phase.

Wei et al. (2005) begin their study by first outlining the benefits to be gained from an ERP system: integration of systems, inherent best practices, and the flexibility to meet the demands of various organisations. However, they also mention the drawbacks and risks, noting that implementation failure occurs in two-thirds of all implementations (Robey et al., 2002). Wei et al. attribute the high cost of implementation and the high failure rate to the fact that implementation requires mutual adaptation by both software and company (Soh et. al., 2003). Various researchers have suggested that resolving such misalignments requires “combining both ERP customisation and organisational change” (Hong & Kim, 2002; Luo & Strong, 2004).

Misalignments can consist of “the incompatibilities between organizational requirements and ERP software in terms of data, process, and output” (Soh et al., 2000), or in terms of “opposing structural forces between an ERP system and the implementing organization" (Soh et al., 2003). Wei et al. themselves describe misalignments as arising "from company-specific, sector-specific, or country-specific requirements that an ERP package does not support and can be clustered into data, process, and output" (Soh et al., 2000). But Wei et al. (2005) state that at the time of writing, there had been "limited research concerning ERP misalignment problems".

In their final conclusion, Wei et al. (2005) offer some advice to managers considering ERP implementation. They recommend managing "misalignment and change at project initialization" as much as possible, and remembering that misalignments and "change actions" can have "potential cascading effects". In the case study by Wei et al., ElectronicCo followed a conservative approach to adopting the ERP technology and tried to minimise the technological change involved. However, this did not entirely pay off, as after ERP implementation, users of the system still demanded "organizational and technology changes". The main finding of Wei et al.'s study is that different misalignment problems will be encountered within different phases of ERP implementation. However, they acknowledge that "further study is needed to identify other factors that may also influence the choice of resolution strategy".

Similarly, Sia & Soh (2007) identify two contexts in which packaged software misalignments of packaged software and clients' business requirements occur. Problems can be caused by both imposed context and voluntarily acquired context. Imposed context relates to contextual details that are country specific, such as the socio-political system, economic structure, or cultural practices unique to a country. Meanwhile, voluntarily acquired context involves differences in processes, or operations due to a company's choice to focus on specific tasks or elements of business. These could include focusing on developing services for niche markets, adopting specific routines for managing critical resources, quests to enhance customer service, the tolerance for management risk, and attempting to match user preferences. Sia & Soh (2007) identify the types of misalignments in packaged software by using Bunge–Wand–Weber elements of ontological structure (deep and surface structure).

Sia & Soh (2007) identify deep structure as the meaning of the core of the real-world system that the information system is intended to model. The real world is made up of things (e.g., fields, persons, artifacts, and social systems) and these things possess properties (e.g., characteristics attached to things) existing at certain states (e.g., ranges of values). The states of things change through transformations (e.g., business rules or laws that define allowable operations). An accounting system, for example, represents the properties of banks and debtors' accounts (e.g., current or saving accounts, transaction currencies), the states of these accounts (e.g., outstanding amounts or balances), and transformations to these accounts. As it does so, it reflects deep-structure characteristics, since it indicates how the economic wealth of organisations and individuals in the real world alters as contracts are executed. These deep structural elements of things, properties, states, and transformations are core elements of structures; the absence of such elements leads to major system deficiencies. For example, missing properties within a system may lead to the system's inability to relate one thing to another.

Surface structure, on the other hand, is identified by Sia & Soh (2007) as being concerned with how real-world meanings are conveyed through the interface between the information system and its users (e.g., through interactive dialogue and reports). Surface-structure misalignments arise when the way that users in the real world access information, input information, and view information on screen and in hardcopy reports differs from the interface provided by the package. For example, customer service officers may need to see information about a client's current transaction as well as summarised information relating to past transactions on the same screen when interacting with the client. Problems may arise if the package screen interfaces are instead designed so that the information is spread across several different screens, as this would make such overviews too difficult for the customer service officers.

Meanwhile, Sia & Soh (2007) also identify four misalignment types. These include: Imposed-Deep, Imposed-Surface, Voluntary-Deep, and Voluntary-Surface. Imposed-Deep refers to there being a missing or inappropriate thing, property, state, or transformation in the system, arising from different country or industry assumptions. Imposed-Surface relates to such things as missing or inappropriate access, input, presentation, or output in the system, and can arise from different country or industry assumptions. Voluntary-Deep relates to there being a missing or inappropriate thing, property, state, or transformation arising from organisation-specific assumptions. Lastly, Voluntary-Surface relates to there being missing or inappropriate access, input, presentation, or output arising from organisation-specific assumptions. Sia & Soh (2007) conducted three different case studies and found over 400 instances of misalignments. Their development of the typology suggested that the most severe misalignments (Imposed-Deep) should result in package customisation, while the least severe (Voluntary-Surface) could usually be resolved through organisational adaptation. The question that could be asked about the issues identified by Sia & Soh (2007) is "what do packaged software companies do to identify misalignments between the functionality offered by the PS and the business needs of a company?"

2.6.3. Packaged Software Implementation 'Customisation'

Light (2005) discusses the customisation of ERP packages after they have been implemented by a company. He notes that customisation of ERP packages is usually explained as resulting from a 'misfit' or 'misalignment' between the functionality offered by the ERP package and the business needs of a company. However, in this study, Light investigates the customisation of ERP packages and delivers some previously unexplained or un-theorised reasons for why customisation may occur.

As Light mentions, despite the commonly perceived benefits offered by standard ERP packages, customisation of ERP packages is still frequent. The customisation and organisational decision-making are not always rational activities, and customisation must therefore be an activity that is "flexibly interpreted" (Light, 2005; Khoo et al., 2011). Light points out that it is incorrect to simply assume that all end-users of a system share a commitment to the development of the system. In fact, these end-users may be affected by various "personal or group agendas" and have different "levels of interest" in the system and varying "degrees of power". Light's argument here is that ERP systems customisation is not "singularly rationally motivated".

The customisation of ERP systems relies on decision-making (Khoo & Robey, 2007; Khoo et al., 2011), and various researchers have criticised the idea that all corporate decisions are made via rational models. Light explains that decision makers may not always have complete information about alternatives and they may often be guided by intuition instead. Moreover, decisions are often affected by a group context and by influences upon an organisation (Khoo & Robey, 2007).

Light (2005) suggests that while previous literature on ERP systems tends to state that customisation will cause trouble, and tends to suggest that customisation is usually carried out in order to add functionality, change the look or feel of the package, or increase the package's efficiency, there may in fact be "other agendas for customisation". For example, an implementation team might use customisation in order to fuel user acceptance of the package. Management staff might suggest customisation so they can keep existing "value adding processes". Light suggests that all of these reasons for customisation are fairly rational, but that there could also be further research conducted into the "lack of reasoning" behind some decisions to customise software. He states that his research is also being developed further as a means of discovering how customisation might aid "sociotechnical integration efforts".

2.6.4. Packaged Software Implementation 'Best Practices'

The concept of "best practice", Yeow & Sia (2008) explain, relates to "a set of proven business methods or an exemplary business scenario" that have been established in past studies of business-related matters and are reflected in the pre-configured arrangement of packaged software (Yeow & Sia, 2008). However, since Yeow & Sia adopt a social constructivist approach throughout their study, they state that the idea of "best practices" is simply a social construction, and that ideas relating to best practice can "evolve subject to negotiation" amongst various groups, even amongst that have competing ideas about best practice (Yeow & Sia, 2008).

Meanwhile, a study by Wagner et al. (2010) delivers a new perspective on the development of large scale packaged software by considering “the processes of mutual adaptation of the technical and social during implementation and maintenance” of packaged software. Wagner et al. (2010) recognise that some misalignments between what a packaged software product delivers and what a company needs or how a company conducts business might only become apparent after the packaged software is implemented. This “post roll-out” phase is the implementation phase that Wagner et al. focus on in their research. Specifically, Wagner et al. examine how an implementation that initially looks like a failure may be turned around so that the information system works. They refer to this phenomenon as “project survival”.

Wagner et al. (2010) suggest that contrary to the idea that it is best to make use of the best practices built into packaged software and to benefit from the way packaged software can help an organisation integrate various business processes, sometimes, in order to ensure project survival (the implementation of packaged software) rather than allow project failure, it might be necessary to continue with some legacy practices. This may be the case even if this makes “migration and future upgrades more difficult” (Wagner et al., 2010; Yeow & Sia, 2008). Wagner et al. state that their findings are “in opposition to the Volkoff et al. (2007) study that found PS changed the relationship between organizational routines and roles by embedding those relationships into the system”. Wagner et al. (2010) state that compromises, such as the decision to retain some legacy practices and features, should be seen as “a necessary characteristic of negotiating practice, not to be viewed as an indication of failure to force change”. PS implementation should therefore be “recast” in the literature about implementation, so that it is no longer regarded as a time of users undergoing a steep learning curve while they must learn to use the new system, but instead regarded as a time of negotiation and further change (Volkoff et al., 2007; Shaul & Tauber, 2012).

2.6.5. Packaged Software Implementation ‘Integration Effects’

One study by Volkoff et al. (2005) involves a 3-year longitudinal case study of a phased PS implementation. One purpose of the case study was to identify integration effects (of both processes and data) prompted by the implementation. Various integration effects were identified during the case study, and Volkoff et al. (2005) consider the effects identified as bearing out the validity of previous work by Thompson (1967) who theorised three types of interdependence: "pooled, sequential, and reciprocal". Using Thompson's interdependence types they discuss the major characteristics of PS-enabled integration and "also identify dimensions of differentiation between business units that contribute to integration problems".

Volkoff et al. (2005) attempt to better define and describe integration, by offering a short definition of integration provided by Lawrence & Lorsch (1967), and expanding upon it. Lawrence & Lorsch suggest that integration is "the quality of the state of collaboration that exists among departments that are required to achieve unity of effort by the demands of the environment". Volkoff et al. (2005) explain that Lawrence & Lorsch see such efforts at integration as often being impeded by the conflicting orientations or goals of different subunits within a company. Therefore, the goal of integration, according to Volkoff et al. (2005) would be to resolve the conflicts that arise from these differences, "without eliminating the differences themselves". One way to achieve integration is by "standardizing work and the data that support it".

Sharif et al. (2005) note that "the multitude of failed ERP implementations and inherent risks involved (McVittie, 2001), has resulted in the emergence of integration approaches such as Enterprise Application Integration (EAI), which seek to integrate information across diverse IS sources (Sharif et al., 2004a)". They refer to the growing externalisation of ERP processes and the sharing of business processes that seek to better integrate processes such as CRM/B2C with other services such as SCM (Bakht, 2003).

Puschmann & Alt (2005) support the idea of the use of integration approaches for packaged software implementation. They identify the contribution of their work as lying not in creating or delivering any new components, but in showing how these existing components can be combined "in a common architecture". Their study is the first to consider portals from "an inter-organizational architecture perspective" and to create an architecture that involves "all three architecture layers, namely presentation, application functionality and data". At the end of their study, they stress that the integration architecture they discuss could be validated or refined further in future empirical projects.

2.6.6. Selection of Packaged Software

One factor leading to the successful use of computing software within a company is how closely involved the manager or owner is with the development of computing resources within the company. The studies cited by Chau (1995) suggest that "the biggest advantage of purchasing software is that it provides economies of scale, while reducing the risk of implementation" (Chau, 1995, 73). Chau also mentions how previous studies have identified that the main factors that seem to be in the minds of owners and managers when they select software can be "categorized into three main groups: software, vendor, and the opinions of other concerned parties".

Chau's major findings were that the owners of small businesses and the managers of small businesses actually used differing standards or held differing views when it came to selecting packaged software for use in their business, and that owners of businesses tended to be much more strategic in their choices. The owners of small businesses took more factors into account when selecting packaged software than did managers of similar small businesses, and the factors that owners thought more important tended to be more technical.

Maiden & Ncube (1998), discuss PORE method "requirements acquisition" for a packaged system. They noted that it is not necessary to have a complete requirement specification from a client before developing a packaged system. Instead, it is only necessary to acquire enough information about requirements that engineers can "discriminate between the candidate products, and then use the selected product as a working prototype for more detailed requirements acquisition".

When discussing how to develop such a product, Maiden & Ncube (1998) state that "most methods and tools" used for software engineering "support only systems design and integration" and that there is a lack of methods and tools for requirements acquisition. Maiden & Ncube (1998) therefore propose a method for requirements acquisition. The method they developed is called PORE (procurement-oriented requirements engineering) and it is a "template-based method for requirements acquisition".

Maiden & Ncube (1998) discuss their PORE model, which "integrates techniques for requirements acquisition and product selection with process guidance for choosing and using each technique". The model combines knowledge gained from various disciplines, including knowledge engineering techniques, feature analysis techniques, multicriteria decision making techniques (MCDM), and decision rationale techniques. These different techniques and guidelines are combined into a series of three or more templates that assist requirements acquisition, product modelling, and product selection. The first template provides guidance when acquiring the essential customer requirements and product information necessary to select and reject products based on the requirements the customer has mentioned. The second template helps with selecting or rejecting products in accordance with customer requirements during supplier-led demonstrations. The third template aids with acquiring customer requirements and product information in ways that help with selecting or rejecting products after customer exploration of those products.

Maiden & Ncube describe their PORE model as being organised around the issue of 'compliance': "this compliance is, in essence, a relationship between a problem and a potential solution to that problem. To do this effectively, the requirements engineer must model not only customer requirements but also each software product". They further explain their model by stating that it "draws on the techniques of task modelling from human-computer interaction, functional modelling from computer engineering, and architecture modelling from system design to model a software product at these three levels".

Maiden & Ncube (1998) argue that “requirements must be as measurable as possible to enable effective product selection” and that engineers should “use software prototypes to aid generation of test cases for product evaluation”.

Muscatello et al. (2003) stress the importance of engaging in reengineering processes prior to selecting an ERP system, basing the selection of the ERP system on the requirements that appear after the reengineering process, and carrying out a thorough “needs assessment” which must cover both software and hardware requirements. Companies should also survey the level of IT skills and knowledge related to ERP within the company prior to implementing an ERP system, so that appropriate education can be supplied, or so a less sophisticated system can be adopted if necessary.

Meanwhile, in another study by Kato et al. (2003) a new method of requirements elicitation termed PAORE (Package Oriented Requirements Elicitation) is demonstrated. Kato et al. (2003) suggest that this method can be used in ERP, CRM, and SCM, and they test the method by applying it to a web-based sales supporting system. Kato et al. (2003) state that their method was developed by observing how experienced analysts elicit and clarify the requirements of their clients. This method has two sub-processes, which are “package selection” and “requirements evolution”.

At the beginning of their study, Kato et al. (2003) state that there is a need to create a method for requirements analysis that can be used successfully even by a software engineer without domain knowledge or without previous software development experience. Their methodology aims mostly at helping software engineers who lack domain knowledge, since they believe that it is only experienced software engineers who are particularly good at eliciting software requirements with regard to the software domain. The PAORE methodology they provide involves the software engineer building up their domain knowledge by “investigating specifications of software packages in advance”, particularly those that seem to meet many of the requirements of the client, and then following this up by eliciting requirements in detail “by both showing the concrete specifications of the selected packages to his customer, and adding the customer's specific requirements" which don't already appear in those specifications.

Kato et al. (2003) suggest that domain knowledge is built up by the software engineer by gathering information about different packages and then comparing the functional features in each package. The specifications of each package can be "systematized as total domain knowledge". They also mention that within PAORE, software engineers can engage in PSM (Package Solution Mapping). This step consists of creating tables which compare packages by listing lines of packages and then creating columns which correspond to features of the packages. They state that by comparing packages impartially in this way, "we can decrease dogmatic dependence on a specific package" – such dependence can, in any case, get in the way of accurate requirements elicitation. This method allows features to be classified into different levels of abstraction, and allows a software engineer's experiences with different packages to be combined.

Requirements elicitation using PAORE first involves the analyst making "an initial list of requirements items" which are obtained through some form of requirements elicitation such as an interview. The output of the PAORE process is an evolved version of this list. The analyst will then use the evolved requirements list to create a Software Requirements Specification (SRS). During the whole process, the analyst should also carry out Future Requirements Mapping (FRM). As the process is completed, the initial requirements list will evolve into a more detailed requirements list. Clients can also be helped through the use of Package Solution Mapping which can make it apparent which package has the requirements that the customer wants.

Kato et al. (2003) suggest that "function oriented requirements elicitation is more efficient than other approaches". However, they also state that the PAORE methodology will only be effective in the case that certain factors in the environment are true. For example, PAORE can only be used on the condition that various packages in the domain have comprehensive user manuals, that there is a domain term dictionary, and that the analysts have some prior experience in requirements analysis. However, one benefit of PAORE is that requirements may be obtained gradually; the customer does not have to supply a complete requirements list. PAORE also suggests ways of using packages themselves as knowledge sources or as sources for a domain dictionary.

In another study of the selection of packaged software, Damsgaard & Karlsbjerg (2010) offer an historical overview of the trend toward standardisation in the development of software packages: the trend toward software being sold to multiple customers, rather than customers developing or seeking a unique solution for their organisation. In their study, they provide seven guiding principles for selecting the most appropriate software package, with the first principle they mention being the most important: (1) when you buy packaged software you join its network; (2) take a long-term perspective of what software you are buying; (3) when choosing packaged software, there is safety in numbers; (4) organisations should consider the presence of open standards and make choices that will help to preserve the open standards; (5) choose a software package with accessible knowledge; (6) choose a package based on the type of standardisation best suited to the organisation; and (7) organisations should not adopt a "wait and see" approach toward selecting software, but instead actively choose a package sooner rather than later.

The first, most important principle delivered by Damsgaard & Karlsbjerg (2010) is that "when you buy packaged software you join its network". This means that when you invest in particular software you become part of a virtual community of all of the users who use the software. The community has an assumed common interest in ensuring that the package succeeds and that the package will continue to evolve, as this would protect their own investments as well and ensure that the time they spend training personnel on using the package is not wasted. The extended network for the product includes vendors, people who make compatible software products, and government authorities who might make rulings relating to the package. The best step to take, suggest Damsgaard & Karlsbjerg, is "to choose to participate in the network that is perceived to provide the best long-term benefits".

The second principle is to take a long-term perspective of what software you are buying. This can be done by imagining how the software or its connectedness to other software may evolve, or by considering the extent to which the software may be adopted by other users. Organisations must “envision a more complex and connected future, or else they risk implementing tomorrow’s legacy systems”. They repeat the advice offered by Shapiro & Varian’s study (1999) that when choosing packaged software one must look forward, but also suggest a need to look back at the past to consider how that product and its network evolved.

In conclusion, the study by Damsgaard & Karlsbjerg (2010) offers those thinking of buying packaged software a guideline for its selection that goes beyond simply considering factors such as “price and immediate features”. The seven principles they discuss encourage a more “multilateral view of software packages”. One major message they communicate is that although an organisation should select the package that best suits their current situation, they must also consider the ‘network’ for that product and whether other companies’ actions or any future technological advancements may affect the future of that package. The researchers suggest that the principles they provided may be an important reference tool for IT managers and may help them to “ensure that vital aspects of the software package acquisition process have not been left out” (Damsgaard & Karlsbjerg, 2010).

2.7. Packaged Software Implementation in SMEs

Haddara & Zach (2011) review the existing literature that relates to the adoption and running of ERP systems in SMEs. Noting that ERP systems have now been almost universally adopted by large organisations, Haddara & Zach (2011) state that ERP vendors have now begun to turn their attention to small-medium sized organisations (SMEs). While ERP systems may be of benefit to SMEs, “the risks of adopting an ERP system are different for SMEs since SMEs are likely to have limited resources, and have business characteristics that are different from those of large organizations”. Haddara & Zach (2011) shed light on the areas that are lacking in current research into ERP adoption in SMEs, and provide information intended to help “practitioners, suppliers, and SMEs when embarking on ERP projects”. In fact, “SMEs have been recognized as fundamentally different environments compared to large enterprises" (Welsh & White, 1981), yet at the time of Haddara & Zach's study, no reviews had been published of literature that deals with ERP implementations within SMEs (Haddara & Zach, 2011).

Haddara & Zach (2011) state that literature shows that there has been a gradual increase of academic interest in ERP usage within SMEs, and that the most frequent methods employed within research articles on this topic are case studies and surveys. They find that the implementation phase was the most discussed phase in the literature on ERP use in SMEs – a finding that accords with the main discussion topics of literature on ERP systems within larger organisations. However, the adoption decision, the acquisition phase, and the use and maintenance phase are also given reasonable degrees of attention within the literature on ERP use in SMEs. The phases for which literature was very scarce or non-existent are ERP evolution and ERP system retirement (Haddara & Zach, 2011). Moreover, Haddara & Zach (2011) state that only two research papers considered "in-house developed systems" to be a feasible option for SMEs, even though "standard ERP packages could compel rigid structures and inflexibility on niche SMEs". Hence, it is reasonable to assume that the recent literature has paid little attention to RE practices of PS implementation from the perspective of SPSVs.

The literature on implementation issues surveyed by Haddara & Zach (2011) found that "project activities, coordination, and project sponsors (Muscatello et al., 2003), employee behaviour, individual characteristics of ERP project management's team, and organization culture have a great effect on the success of ERP implementations in SMEs (Chien et al., 2007)". One study conducted by Newman & Zhao (2008) investigates the importance of business process modelling and business process re-engineering during implementations carried out in SMEs (Haddara & Zach, 2011). The conclusion of Newman & Zhao's (2008) study is that "in some cases, ERP systems should be customized to fit with niche SMEs and not vice versa, as they might lose their competitive advantage by complying with standard ERP processes".

At the end of Haddara & Zach's (2011) discussions of the literature they reviewed, they make some further comments about the literature and suggest further avenues for study.

First, they suggest that although they found and reviewed 77 articles, this was still a very small number of articles to be published on the topic within 10 years, given the growing importance of ERP systems in relation to SMEs. They believe that "SMEs did not receive appropriate attention in comparison with ERP in LEs". They also identify specific gaps in the literature. These include a lack of studies that look at "ex-ante cost estimation, financial feasibility, and investment evaluation studies of ERP projects", lack of comparison between "SME's-specific ERP and general ERP systems" or between "industry-specific ERP packages vs. general ERP ones". Haddara & Zach (2011) find that very few studies had been made relating to the evolution of ERP systems within SMEs, and no studies had considered the retirement phase of an ERP system in relation to SMEs. Lastly, Haddara & Zach (2011) state that while they did find 77 articles relating to ERP systems within SMEs, most of the SMEs were involved with traditional manufacturing, and it could be beneficial to obtain results pertaining to different types of industries, or if studies relating to ERP system use within SMEs were more explicit about exactly what kinds of manufacturing or industry the SME was involved with. They also note that "some articles examined of ERP implementation in SMEs, however, the differences of ERP implementation methodologies and their impact on ERP projects had scant attention".

They note that most of the studies conducted have considered companies located in America, Australia, Europe, and Asia. There was a shortage of studies investigating SMEs in Africa or in the Middle East. In general, existing literature have adopted a one-sided perspective in terms of data collection. They have focused on the customer's side, whereas other perspectives could enhance the understanding of certain phenomena.

2.7.1. Critical Success Factors for Packaged Software Implementation in SMSs

Like some other researchers in this area, Snider et al. (2009) identify SMEs as facing specific challenges when adopting ERP packages. For example, SMEs may often lack adequate human and financial resources to support such initiatives (McAdam, 2002). Despite this, the rate of ERP adoption at SMEs has been catching up with ERP adoption in large companies.

In their literature review, Snider et al. (2009) briefly discuss the framework provided by Loh & Koh (2004) in their study of CSFs mentioned in previous literature on ERP implementation. Loh & Koh's study divides the applicability of CSFs into three phases of ERP implementation: '"preparation, analysis and design", "implementation", and "maintenance"'. Snider et al. (2009) note that their study builds on the review by Loh & Koh but updates it by considering a few more studies conducted after the year 2000.

Snider et al. (2009) go into further detail about the financial and organisational issues that might create particular difficulties for SMEs wishing to implement ERP software. These include a lack of financial resources for engaging consultants, a lack of staff who are equipped to carry out projects or who are experienced in IT (or even a lack of staff in general), and a lack of financial resources for any extra training that may be needed. Snider et al. (2009) also note that past literature on the topic has found that SMEs often lack long-term planning but that their small management teams may result in efficient decision making (McAdam, 2002). Snider et al. (2009) follow this by briefly mentioning the call in previous studies for investigations into ERP implementation in SMEs. For example, Huin (2004) argued that unless studies were made into the differences between large organisations and SMEs, managing ERP projects would continue to be slow, difficult, and potentially unfruitful for SMEs.

Snider et al. (2009) explain that in the context of their study, the level of success of an ERP implementation relates to "the extent that potential benefits were achieved (Davenport, 1998)", "the costs associated with achieving those benefits, and the duration since going live (Markus et al, 2000a)". Their study identifies various critical success factors for ERP implementation in SMEs through the use of a case study of five companies. The first critical success factor discussed is "operational process discipline". They asked the five companies studied "about documentation and consistency in executing operational processes (i.e. information flows) prior to the implementation. Companies having greater consistency prior to implementation appeared to achieve more successful implementations regardless of the level of documentation. The two unsuccessful cases had good documentation, but low discipline in adhering to standards set in documents". Those two companies appeared to have difficulty adhering to processes that were newly developed by the ERP system. Snider et al. (2009) state that "having inconsistent operational processes conflicts with the procedural rigidity of ERPs" and suggest that some companies may need to carry out process benchmarking and improvements "prior to enforcing standardized procedures brought in by ERP". While SMEs often have a fairly informal kind of environment, the introduction of ERP may make it necessary to have greater operational process discipline.

Snider et al. (2009) also find that the companies who have smaller implementation teams have the more successful implementations. They posit that the large implementation teams studied seemed to have adopted a stance of being isolated from other employees in their company. They did not seek the input of other employees, seeing themselves as the experts selected for the implementation. The smaller teams instead sought user assistance regularly. Larger teams also had more difficulty with reaching a consensus, and each large team also had at least one unreliable member that did not complete tasks. The larger teams also appeared to be more difficult for the team leader to manage or for external consultants to work with.

"Project management capabilities" are also identified by Snider et al. (2009) as a critical success factor for ERP implementation in SMEs. This kind of project management involves "documentation and leadership" related to planning and managing tasks and meeting deadlines. Snider et al. find that the success of implementations often appears to be directly connected to who the project manager is. Those companies who had external consultants in charge of the implementation actually tended to have more success than those whose project leader was from inside the company – probably because the external consultants had more project management experience. Extensive project documentation, setting target dates and holding frequent meetings about the project all appeared to be beneficial components of project management.

"External end-user training" is also identified as a CSF by Snider et al. (2009) and is divided into "training" ("software specific instruction") and "education" ("general skill upgrading"). Within the five companies studied, the success of the end-user training appeared to be linked to whether training was provided by an external consultant or by an internal company employee. The more successful implementations involved using an external consultant for training. In one company that asked their own employees to conduct the training, it was discovered that the training materials created were not detailed enough and that employees from various departments obtained highly varying degrees of training in terms of the hours provided. Snider et al. (2009) therefore suggest that externally-provided training might particularly benefit SMEs.

“Management support” is also identified by Snider et al. (2009) as a major CSF for PSI, consisting of both financial support for a project (including readiness to hire the right suitable consultants) and “encouraging staff toward the implementation” by acting as a project champion. Management support throughout an ERP implementation might be even more important for SMEs than for large companies, because of the close-knit nature of SMEs.

Choosing a “qualified consultant” is also identified by Snider et al. (2009) as a CSF. They suggest that the assessment of a consultant’s quality may be linked to their “business understanding, software knowledge, and soft skills”. Consultants should be knowledgeable about the whole software package, not merely about one module of it. One company studied (Company 4) hired a consultant that was disliked by the employees. The employees therefore avoided speaking with the consultant and tried to solve issues on their own, an approach that was not ideal. Snider et al’s conclusions about what qualities make a good consultant support the findings of Bingi et al. (1999).

Snider et al. (2009) found that four of the five companies they studied chose to modify their ERP software after installation – this despite the frequent recommendation in literature that finding a “proper fit between processes and software” is critical for ERP success. The companies that made these modifications did face some challenges when it came to using and understanding the software after modifications were made. However, making such modifications did not necessarily lead to project failure: two of the companies that modified their software had successful implementations, and two unsuccessful.

They also found that it was not necessarily a deterrent to implementation success if an SME did not have or did not communicate a formal business strategy to its implementation team. In "two of the three successful cases the business strategy was not formalized or communicated to the team". It appeared that all of these companies were actually making short-term plans in response to current business requirements or expected business growth. However, two of the companies acting in this way still derived significant benefits from their implementations.

In conclusion, Snider et al. (2009) emphasise that ERP implementation in SMEs appears to have six main CSFs: "operational process discipline", "small project team", "project management capabilities", "external end-user training", "management support", and "qualified consultant", and that those companies that manage these factors effectively will have a higher chance of implementation success . They state that their findings about modifications creating some technical challenges for the companies "point to the need for implementations to ensure that both technical and business expertise is integrated during software testing". While they agree with previous studies that project management and end-user training could be considered CSFs for ERP implementation in SMEs, they extend these findings by stating that project management and end-user training is best performed by an external consultant.

2.7.2. PS Customisation in SMEs

Zach & Monkvold (2012) investigate the reasons behind and context for customisation of ERP systems in medium-sized enterprises (SMEs). They carry out case studies of four ERP implementations and instances of customisation in four companies in the Czech Republic. Zach & Monkvold's paper adds to the scarce literature on customisation practices in SMEs and features an approach by which the authors consider the contexts influencing customisations prior to "going live" and after "going live" (at two different phases of the ERP life-cycle). The main focus of their study is on "distinguishing influential factors of the SME context".

Zach & Monkvold (2012) note that “the ERP literature includes a number of studies exploring the issue of ERP system customisation” and that many such studies advocate that ERP systems should be implemented with minimal customisation. Despite this, a number of studies have considered why customisations occur. Most of these studies, however, relate to large-scale enterprises. Zach & Monkvold explain that ERP vendors are now turning toward selling ERP systems to medium-sized companies, having developed many midrange and less complex ERP systems. However, companies of this size have their own specific problems with ERP implementations, and ERP system implementation remains a challenge for many SMEs (Malhotra & Temponi, 2010; Olson & Staley, 2012). The limited research into ERP implementation in SMEs shows a tendency for SMEs to favour customisation rather than adapting the company’s business practices (Quiescenti et al., 2006). Zach & Monkvold (2012) therefore state that the core question behind their own research is “why do SMEs seem to favour ERP system customisation?”

Past studies into the phenomena of ERP implementation in SMEs therefore establish the importance of conducting separate research into implementation within the SME context. SMEs are fundamentally different from large enterprises in several aspects and studies of ERP implementation frequently argue that findings from large companies cannot be applied to SMEs (Buonanno et al., 2005; Laukkanen et al., 2007; Mabert et al., 2003). SMEs differ from larger companies in terms of their ownership structure, market orientation, and level of resources that can be applied to IT; they may also have fewer employees with a high level of IT expertise (Zach & Monkvold, 2012). Such differences are likely to influence the factors behind their choices to customize ERP systems.

Zach & Monkvold (2012) discuss previous literature’s findings about the main reasons for customisation. While a functional misfit between the functionality offered by an ERP system and the business needs of the organisation is the most cited reason, Zach & Monkvold (2012) notice that studies by Light (2005) and by Rothenberger & Srite (2009) provide further reasons, which generally revolve around the organisation’s business culture, resistance to change, lack of knowledge about a product, or fear of personal disadvantage resulting from the change. Meanwhile, some implementation teams may also be perhaps overly willing to accommodate requests for customisations.

Again, however, these studies by Light (2005) and by Rothenberger & Srite (2009) actually relate to implementations in large organisations. The literature pertaining specifically to ERP implementations in SMEs stress the importance that SMEs place on system flexibility. The high level of customisation carried out by SMEs is linked, within this literature, to this desire for flexibility (Zach & Monkvold, 2012). Medium-sized businesses appear particularly determined to maintain their "unique business processes". Adaptability and flexibility is deemed so important that Olsen & Saetre (2007a, b) "proposed that in-house development of ERP is the best alternative for many SMEs".

Zach & Monkvold (2012) identify the main reasons behind the customisations carried out by the four organisations, dividing the reasons into those related to the pre-"going live" time and post-"going live", although some of the reasons involved are relevant to both of these phases. In their discussion of the "prior to 'going live'" phase, they identify "resistance to change" as the primary reason behind customisations. All of the organisations studied had decided that they preferred to make the ERP system suit their organisation than for them to change their organisation's processes to suit the ERP system. However, on further consideration, Zach & Monkvold realised that the real reason behind this resistance was a wish to preserve "unique business processes". Each company had idiosyncratic business processes and considered those processes to be essential to the future functioning of their company. Another reason for some of the customisations was "functional misfit". Again, however, these misfits between what the ERP system offered and how the company actually ran occurred largely because of these idiosyncratic business processes that the company wished to protect. In fact, all of the organisations studied had functional misfits with the pricing mechanisms offered by their ERP system because they had their own special pricing policies.

The ownership type of a company may also be a factor influencing customisation (Light, 2005). For example, Zach & Monkvold (2012) found that in SMEs (1) the decision to implement an ERP system tended to be made by only one or two people, who asserted considerable personal control over the direction of the organisation, and (2) in these medium-sized organisations, the decisions to implement ERP systems were generally motivated only by a wish to replace unsatisfactory legacy systems, not to change any of the organisation's processes. The implementations were therefore driven only by technical considerations and not by any long-term strategic motivation.

After the "going-live" date, Zach & Monkvold discovered that the reasons behind customisations changed. Some of the customisations at this stage were prompted by the growth of the organisations and the dynamic nature of their business activities. However, this did not mean that the organisations were changing their business processes to change the ERP systems, rather that they began adding further functionalities to the ERP systems. The stage of growth the organisations were going through therefore affected which customisations they made after implementation. In some cases, the organisation realized that their ERP system was not particularly mature, so they wished for further functionality. Zach & Monkvold (2012) therefore argue that "the maturity level of the selected ERP systems required a high level of customisation".

Since SMEs generally thrive because they have successfully done something unique within a niche market, they may seek to protect that competitive advantage by avoiding any standardisation encouraged by an ERP system. Thus, they prefer to make customisations. This idea corroborates the findings of previous studies (Bernroider & Koch, 2001; Quiescenti et al., 2006; Snider et al., 2009). However, Zach & Monkvold (2012) note the level of influence held by the owner or CEO (often the same person) within SMEs, something which is a new finding. If the CEO or owner of a company resists change and instead desires customisation, this is difficult to oppose. Another new finding by Zach & Monkvold (2012) is that customisations are more likely within SMEs because their ERP implementations are more likely to be driven only by technical concerns, not by strategic ones. They note that while previous studies, such as those by

Robey et al. (2002) and Rothenberger & Srite (2009) had found that those large companies who are resistant to change are more likely to engage in customisations, “this lack of strategic motivation is more frequent in SMEs”. The limited knowledge of IT or limited experience with ERP systems within SMEs might lead to increased customisation levels, while the continually growing nature of some SMEs is another factor behind customisation. Meanwhile, the maturity level of the ERP system is another factor behind customisation. The domestic ERP systems selected by the four organisations studied were less sophisticated than a standard SAP system. The organisations therefore would have found that the ERP systems selected did not offer all of the functionality they required, so they later added the functionalities needed via customisation therefore, that there is a need to understand the process that SPSVs apply in order to identify misfits between the PS functionalities and SMSs’ business process.

2.8. Summary

The development of an information system involves knowing what to create and how to create it. Understanding what an information system needs to do demands that analysts determine clients’ business requirements.

Packaged software is often developed in several consecutive releases and there is great competition within the packaged software industry and between different packages. This is one of the elements specific to packaged software, and contributes part of the reason why the characteristics of packaged software Requirement Engineering (RE) differ a great deal from the characteristics of bespoke RE. With packaged software, there is no distinct and defined set of users. Instead, there are potential users, an imagined group of people who may fit the profile of the intended product user. Eliciting requirements from this group of users and customers is one of the activities that distinguish packaged software RE from bespoke RE. The elicitation of such requirements is mainly managed through marketing, technical support, user groups and trade publication reviewers. Therefore, it seems reasonable to assume that not all packaged software will be suitable for clients’ business requirements and that conflict will occur between the functionalities provided by packaged software and the business requirements of clients, once software companies implement packaged software.

Implementing packaged software can be a major project requiring a significant level of resources, commitment and changes throughout the implementing organisation. Often the implementation of PS is the single biggest project that an organisation has ever launched. The stakes are raised when one considers that in the past, some cases of failed implementations have led to the demise of various companies. Issues surrounding the implementation process for packaged software have been among the enduring concerns in industry. Studies of PS implementation at SMEs therefore demonstrate that the particular context of SMEs certainly has an effect on the level and type of misfits between the PS functionalities and customisations carried out. I suggest, therefore, that there is a need to understand the process that SPSVs apply in order to identify misfits between the PS functionalities and SMEs' business process.

In order to better understand the implementation process for packaged software, it would be beneficial to study the phenomena of RE for packaged software implementation (referred to as PSIRE in this thesis), and in particular, to observe PSIRE in practice. The remaining chapters of this thesis therefore provide details relating to my ethnographic study of PSIRE as practiced by analysts in small software development companies (SPSVs), outlining my research approach, providing the results of the research, and discussing the results. In order to study PSIRE in practice, I conducted field work in which, following an ethnographic methodology, I observed analysts at 2 SPSVs located in Jordan. The study and my interpretation of the data collected follows a qualitative research approach. The data obtained was collected by means of participant observation, unstructured interviews, informal interviews, and focus groups. In Chapter 3 I provide detailed discussion of the research approach, research methodology, ethnographic setting, the participants in the study, and the data collection methods used.

Chapter 3 Research Approach

3.1. Introduction

The purpose of the present study is to understand the phenomenon of packaged software implementation requirements engineering (PSIRE). Many researchers in the field of software engineering and information systems suggest that the best way to understand the phenomenon "packaged software implementation requirements engineering" practices is to observe and interpret the experiences of the participants involved in the process. Two groups of participants in information system projects are analysts and users. These analysts and users may possibly understand and agree among themselves upon the requirements of a system. This study focuses particularly on analysts' perspectives regarding the requirements engineering practices they follow as a part of PSIRE.

This research is organised along the lines of the structure used by Myers & Avison (2002). Such a structure provides a systematic way of describing and defining the research approach. The structure is as follows: Section 3.2 gives a general overview of the research from a philosophical point of view; Section 3.3 discusses the research method; Sections 3.4 to 3.10 give an overview of which data collection methods were used during this research and how they were applied; Section 3.11 provides a description of various modes of analysing and interpreting the data used within this research.

3.2. Research Philosophy

The major research question I ask and explore within this thesis is "How is requirements engineering in packaged software implementation contexts enacted at SPSVs?" This question could be answered by observing the actions of members of an analysts' team as they conduct PSIRE. This is a major aspect of the research phenomena is that requirements engineering are an essential set of activities for gathering users' needs. An interpretive philosophy is a highly appropriate approach to use here, as it can allow the researcher to capture the views of the participants, and can assist with understanding the phenomenon of packaged software implementation requirements engineering (Trauth, 2001). My research approach can be categorised as interpretive research with the goal of empirical investigation of PSIRE (Klein & Myers, 2001).

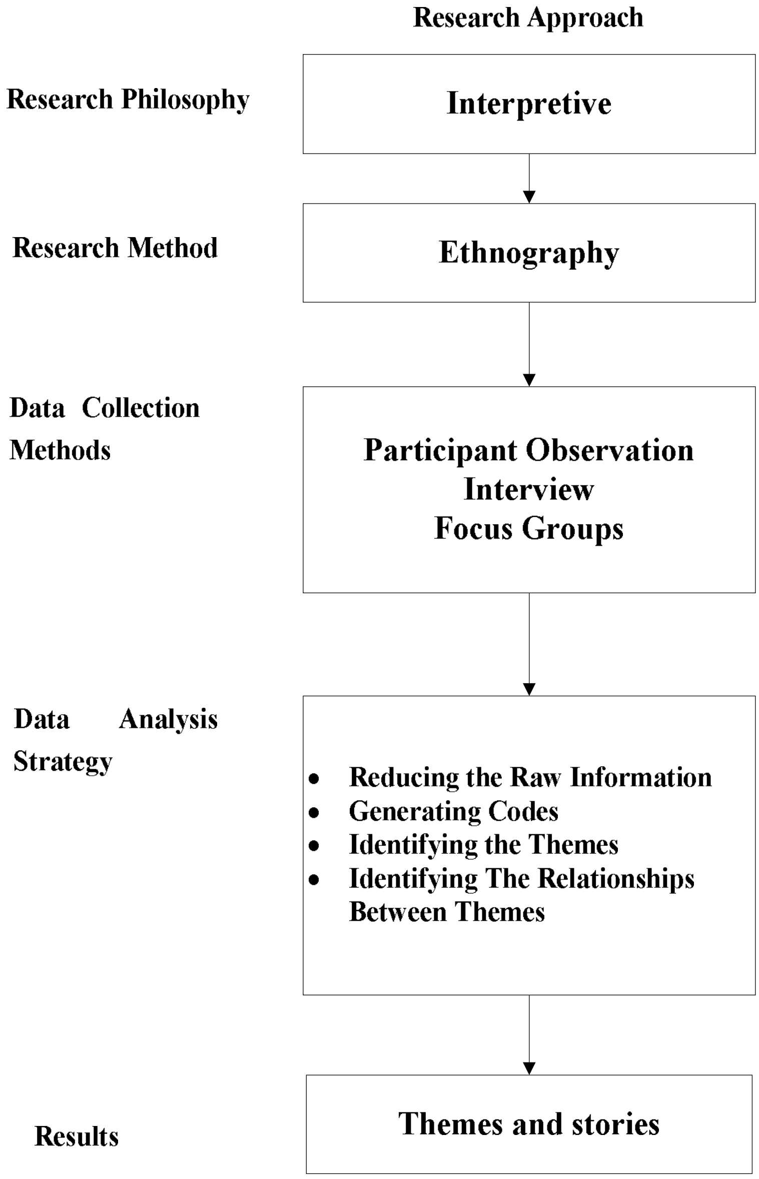


Figure 3.2-1 Research Approach

When choosing a particular research philosophy, certain assumptions and perspectives are accepted, and certain strategies and interpretations should be involved. Revealing what is hidden behind the “facts” shared by the stakeholders in RE can be done by applying an interpretive approach to their interaction process. This creates a more in-depth look at the context of the interaction. An interpretive approach also reveals what participants shared, and how, when and why sharing was done in a particular way (Myers, 1999). For the purposes of this study, I decided that utilising an interpretive philosophy to investigate the data acquired would be the most suitable approach. I realised that the interpretive philosophy would be able to generate new understandings of the complex phenomena that influence PSIRE. Figure 3.2-1 presents an overview of the research approach.

3.3. Design of the Study

A research methodology is a strategy of inquiry which includes research design and data collection (Myers & Avison, 2002). The choice of research methodology influences the way a researcher collects data. Specific research methods also imply different skills, assumptions and research practices. Some research methods in the information system area include action research, case study, grounded theory, and ethnographic research (Myers & Avison, 2002).

Action research brings refinement or new facts to already existing evidence from previous studies, through the collaboration of the researcher and the participants (Collis & Hussey, 2009). Due to the nature of my research that focused on observing events in natural settings, and which had an absence of collaborative intervention, the action research method was not chosen for the study.

The case study is a research strategy which focuses on understanding the research hypothesis and theory. It aims to provide description, to test theory, or to generate theory. Typically the sources for data collection are archives, interviews, questionnaires, and observations (Eisenhardt, 1989). Considering this research method, the researcher should have a preliminary hypothesis or theory to test, contextualised in specific time and space frames (Yin, 2008, p. 27). Since the particular study I discuss here lacked a theoretical framework (Yin, 2008, p. 28), research boundaries (Collis & Hussey, 2009,

p. 70), and the context of a particular phenomenon, since these were factors which would make the understanding of users' requirements problematic, the case study method was removed from the list of potential research methods. However, the evolution of this study might lead to the development of such factors.

Grounded theory is a qualitative research approach that was originally developed by Glaser and Strauss in the 1960s. It is defined as "a research method that seeks to develop theory that is grounded in data systematically gathered and analysed" (Myers & Avison, 2002, p.9). According to Urquhart (2001), grounded theory formulates clear features of a topic and provides a foundation for the conceptual idea based on the organisational context. The goal of grounded theory is to find relevant evidence that would be precise, thorough, and capable of replication, which would lead to its consistency with empirical observations (Eisenhardt, 1989; Orlikowski, 1993). The grounded theory method was not chosen for the study because the main concern of this study is to understand the phenomena of PSIRE from the viewpoint of analysts within the field, and to understand how analysts actually practice PSIRE. It was decided that grounded theory would not be the most suitable method for this study since grounded theory generally relies very heavily on the collection of data through interviews. My aim, however, was not merely to collect information about what analysts said they do as part of PSIRE, but to witness what they actually do in the field. I felt that by observing what the analysts actually do in the field and to what degree it matches with what they say about PSIRE, I would obtain further context for my study and a richer range of data. For this reason, I chose to follow an ethnographic research method.

Ethnography is a research method well acknowledged and widely used in sociology. The main purpose of ethnographic research is to describe people collectively, drawing attention to social and communal ways of life and to behaviour and customs (Myers, 1999). It is an appropriate method for studying social interactions, behaviours, beliefs, and perceptions that occur within groups and organisations but are not yet clearly understood. Information about the social life of a group can be studied through an ethnographer's immersion into the life of people from that organisation (Hammersley &

Atkinson 1995). The central aim of ethnographic research is to provide insights into people's views and actions by collecting information through interviews and observations. This research method can incorporate multiple perspectives into the research phenomenon (Holzblatt & Beyer, 1993). Therefore, ethnographic research was chosen as the most suitable research method for this study.

At its core, the ethnographic approach relies not only on observation of social actions and interactions in natural settings ("fieldwork"), but largely on close-up experience and the participation of the observer in the personalised settings of an organisation or community. Such research requires long-term participation in order to obtain a portrait of the group under study. The "unstructured data" that is collected (data that is not coded at the time of data collection) allows the researcher to carry out inductive research, rather than test hypotheses. Researchers who use ethnographic methods must pay careful attention to the process of field research in order to identify patterns from observations and interviews. At the end of the research, the researcher will have a wide range of qualitative and quantitative data obtained through learning and testing different problems from different perspectives. Hypotheses can then be made from these different sets of data. The collected data can be compiled in charts, tables, and graphs, but the ethnographic report is always presented in narrative form, with an introduction, setting of the scene, analysis and the conclusion.

Due to its naturalism, lack of time and space constraints, and bottom-up nature, an ethnographic approach was chosen as the most appropriate method for the current study. Although ethnographic research has its roots in anthropology and social studies, it is also well accepted in IS, as long as it brings new contributions to the understanding of phenomena in IS (Harvey & Myers, 1995; Myers, 1999).

According to Myers (1999) using the ethnographic method puts the researcher in a culturally challenging situation because he/she has to have direct interaction with the participants being observed, in a certain and sometimes unfamiliar environment. Fortunately, I was provided with the opportunity to conduct my study in my own country where I am well aware of the life-style, the language, and the culture. Being familiar with the culture, however, did not mean that I was not intruding in an orderly organisation in which people would have their own perspectives and assumptions. The choice of how to conduct field work is the first step towards understanding the way an organisation works, and their perspective on how and why it functions a certain way (Harvey & Myers, 1995; Myers, 1999). Field work methods gave me a chance to observe participants in natural settings and to find out the views of the participants in the domain observed. The participants were also invited to comment on the conclusions and interpretations of the data (Chapter 4 & 6), which contributed further to the clarification of my understanding of the phenomena under study.

In summary, ethnographic research has the potential to provide great depth for any researcher that wishes to understand a phenomenon in reality. Not only does it demonstrate and describe various people's actions, it can also have a comparative aspect, indicating whether what participants say about their aims and approaches matches with what they do (Myers, 1999). By comparing what people say and what people do, and by observing how people practice what they say they do, a researcher can clarify understanding of a particular phenomenon or practice. The researcher can reach an in-depth understanding of the group under study and the broader context in which the study took place through gaining an intimate familiarity with the everyday life of the participants. The research also provides an insight into what we take for granted and makes the researcher note the specificities of the process within the group under study.

3.3.1. Getting Access to a Setting for Field Work

One major challenge associated with ethnography is getting access to a field work setting that can accommodate the researcher for a sufficiently lengthy period, in this case several months. The following section will explain the attempts I made to gain access to opportunities to conduct field work. I would like to share my personal experience of getting access to field work settings during my study, as this might be useful for other researchers who should be aware of the challenges that I faced.

Starting in New Zealand in December 2010 I applied for ethical approval for my research method and data collection approach. In the same month my application was accepted by the AUT Ethics Committee, which gave me a chance to start looking for participants for my study. I started by sending emails and calling software development companies in New Zealand to explain my research idea. Eventually I received some responses from these companies, with the condition that I would interview the analysts rather than participate in the field work.

After accepting the companies' offers to interview analysts, I interviewed 8 analysts at three different companies. Staying in New Zealand in 2011 and struggling to find field work research opportunities pushed me to start thinking "outside the box" in order to find participants that could help in the research. Therefore, I discussed with my supervisors the possibility of using undergraduate students in a final project paper at Auckland University of Technology where the students who enrolled were supposed to develop software that met the clients' needs. In this case, interaction would occur between students who would be acting as analysts with the client giving software requirements to them. The plan was to apply for ethical approval to be accepted and start finding students who would accept to be part of this research from December 2011. However, the timeframe involved with my waiting to be accepted to use students as participants would last for at least 6 months, and it would take some time to find suitable projects that would fit with my research. While waiting for the approval to use Auckland University of Technology students as participants my supervisors and I discussed the possibility of my finding participants in home country Jordan. Since it had proved difficult to find suitable participants in New Zealand, the agreement with the supervisors was that data could be collected from overseas companies as well, since it would not go against the research phenomena.

In November 2011, I decided to go to Jordan for a few months to search for participants from software development companies. Being from the same cultural and linguistic background aided my search for participants for my research.

After I arrived in Jordan in late November 2011, I started looking for participants by sending emails, calling, and holding personal meetings with company managers. I conducted 4 interviews in the first month but I believed that I needed companies to allow me to be in the field, in order to obtain a better understanding of the phenomena under investigation.

I met the general manager of Organisation 1, a company that develops business software solutions, and explained to him the research idea and the expected results from the research. The manager showed strong interest in the idea and discussed some concepts and possible ways to use the results for his own company in order to improve the PSI process. The manager suggested I make a presentation about the research idea for analysts in order to give the employees a clear vision of the research area and objectives, so the employees could make the decision whether to be part of the research or not, without the manager forcing their participation.

I had prepared a presentation about the research idea. The presentation took place in one of the company's meeting rooms. I explained the research idea and the anticipated results to the analysts and pinpointed how these results might benefit their own skills and performance. The participants' information sheet (Appendix A) was provided during the presentation, and the employees were given two weeks to think about whether to accept or reject participating in my research. After ten days, I received a call from the company advising me that I was accepted to participate in the field work for my research.

The other company that agreed to participate in my research is Organisation 2. During my search for participants I met one of my colleagues who had studied with me during my undergraduate degree and I explained to him my research idea. He expressed his interest in helping me find participants. He made a call to Organisation 2's software manager and arranged a meeting with him. I met with the software manager and explained to him the research idea and he asked me to present it to the general manager and the rest of the software department team members. Following the same procedure as

I adopted with Organisation 1, two weeks were given to the company to decide whether to accept my invitation (they did accept my invitation; see Appendix A).

3.3.2. An Ethnographic Setting

In this section I present one of the two cases I conducted in an ethnographic setting. The case discussed here is the case of Organisation 1. This life experience is truly the best thing I have done to date. Although at the beginning I did feel a bit bored and excluded, with time I gained the trust of the companies' employees, made new friends, and was soon involved in the company life, which made me feel like part of the organisation. This experience gave me access not only to the lives of the companies' employees within the office walls but also made me involved in the analysts' lives outside the work environment. In Amman city during winter time, although the temperature outside was close to zero with strong winds and snow, I felt warm and accepted by the two companies.

Figure 3.3.2-1 winter in Amman

Figure 3.3.2-2 Winter in Amman- the researcher

My ultimate goal for the research project was to gain insight into the organisations and to observe, be part of, and describe the daily practice of the software analysts in the companies. I wanted to immerse myself into the companies' lives in order to be able to show what is happening behind the walls of the packaged software producers.

> *In people's daily lives for an extended period of time, watching what happens, listening to what is said, asking questions - in fact, collecting whatever data are available to throw light on the issues that are the focus of the research.*
> *[Hammersley & Atkinson 1995, P1]*

To retain the true atmosphere of the setting and show the spirit of the interviewees, the data, which includes interviews, emails, as well as field notes, was not corrected for grammar and the like. The original raw data expresses the way the participants were talking. Only orthographic errors were corrected in this work. Since most analysts had English as their second or third language, their quotes presented in this work show the way non-native English speakers express themselves in English. The errors that can be found in my notes show the speed in which the data collection took place. Although most of the interviews and meetings were in English, some of the interviews were, however, in Arabic, and some of my notes were originally in Arabic. These interviews and notes, however, were translated into English later on, during the data analysis stage. This was done through an official translator in New Zealand (see Appendix F). All this kept the spirit of the organisations' work and the employees' perspectives on their work activities intact and revealed situations that happened in these specific settings.

3.3.3. Arriving at the Companies

Although I am originally from Jordan, I have been away from the country since 2006. Moreover, all my life I was living 80 kilometres away from the capital city, so for me Amman was like any other foreign town. I found it hard to orientate myself without a map and sometimes looked like a tourist, asking strangers for directions. Luckily, I have a brother who relocated with his family and is running his own business in Amman, so he was a good guide for me around the city and its suburbs. On the day when I first went to Organisation 1, my brother guided me towards a big multi-level office building in an industrial area situated within walking distance from the city centre. We arrived at the office building around 8.30am. It was a cold cloudy morning but I had to stay outside for about half an hour since I arrived a bit early and had an appointment only at 9am. When the glass door closed behind me I found myself in a warm huge reception area, where I asked the receptionist to inform the manager that I had arrived. I had to wait at the reception for a while before the manager was ready to meet me. While seated and waiting I looked around the reception area and noticed that there were five computer desks with women behind them, apparently doing administrative work. After what seemed like an eternity to me, a man came out to meet me. It was the manager, who welcomed me with a handshake and summoned me to a meeting room. He explained to me that, as a researcher, I could attend meeting sessions with the team leaders. The manager agreed that I could start right away, and, after our short conversation was over, he led me into an office with a desk in the centre.

Everything seemed to be very different from what I had become used to seeing in New Zealand. First of all, I realised that the dress code in the company was very different from what I had become used to wearing in New Zealand. The passion for smart clothes that I had had when living in Jordan had changed to my liking jeans and casual clothes. Here I realise that unlike me, everyone in the company was dressed up in business suits and ties, looking very smart. After spending an hour in my new office, a man appeared at my door, introducing himself as one of the analysts' team leaders and inviting me for a tour around the entire office. On the way around the office, I was introduced to several other team members (6 members) and was told how and where to find them in the office for any further communication.

When I went back to my office after the quick introduction to other team members, I intentionally left my office door open. With my desk in the centre of the room facing towards the open door, I could see everything that was going on outside my office door. My intention in doing this was to be able to casually make eye contact with passers-by, and to cast an occasional smile. People were passing by to grab drinks, go to the bathroom, and coming and going to and from their small offices. My friendly appearance made some of them initiate casual chats with me about my position in the company and other information related to my appearance there. This quickly allowed me to be known by other people in the organisation. It also enabled me to find out a bit more about other members of the company and whether I would be working alongside them or not.

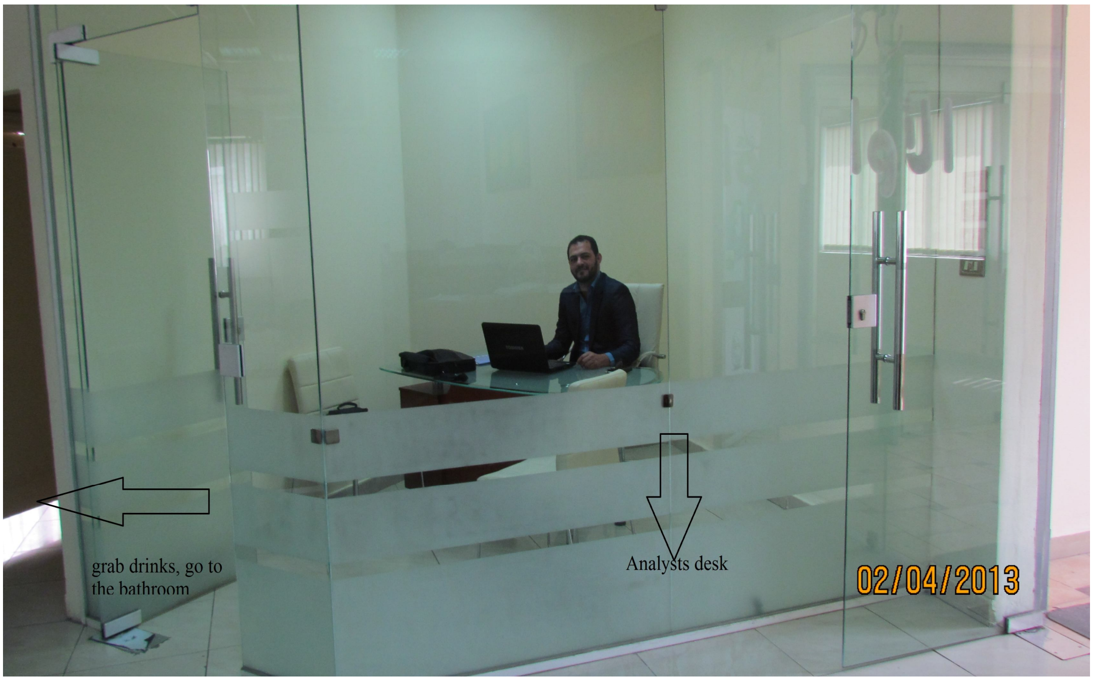


Figure 3.3.3-1 my office

As you can see in Figures 3.3.3-2 & 3.3.4-1, the company had a formal dress code. This, of course, made me change back to wearing suits every day in order to show solidarity with them and become one of the company members. Since suits were the company’s dress code, no one was wearing jeans and a casual shirt. When I brought up the question of dress code with my manager, he explained that the formal dress code allowed the company employees to show their professionalism as well as express the company’s style of work. I was not really surprised about the dress code, to be honest, because I knew that software companies in Jordan are known for their formality, but with time I must have forgotten this, and therefore attended work in jeans and a shirt on my first day. It was a good reminder for me as I was to go through a similar experience when meeting the manager in the second software development company where I had been accepted to do research.

Figure 3.3.3-2 analysts & developers team

When I arrived at the second organisation I was dressed up smartly, wearing a suit and a tie. To my understanding, this made a different impression on the manager. I believe this also made it easier for me to be accepted by other employees. Since I was not looking different than them, I blended into the environment, as though I were one of them.

3.3.4. The Atmosphere at the Companies

I found the atmosphere among the employees very friendly and generous in both organisations. People casually helped each other with work-related issues as well as with other situations happening in people's private lives outside the office walls. Even if a difficult situation happened at the last minute, they would help their colleagues, for example, with taking analysts to the clients, or, like in my case, helping me find a new apartment and even relocating and moving in.

Figure 3.3.4-1 The Analysts & Developers team

When I expressed my surprise about the employees' relationships to one of the team leaders at organisation one, I was advised that "...most people in the company are friends and so, everybody knows each other for at least three or more years, and some even are family members". Secondly, I was told that software is an artefact built by many, and "...if people do not work together, don't follow the same goals, don't comply with the company's mission, the software cannot evolve".

3.3.5. Introducing People

Ethnography is all about people. All the people I met during my work experiences in Jordan, each member of the organisations, were important to my study in one way or another. They all provided me with insights about the work process and shaped my point of view. The employees of the organisations were not my informants, but rather individuals and colleagues who allowed me to become part of their lives, and some of them became good friends with me.

There were nine team leaders in the two companies that I was fortunate enough to work with. All of them were helpful and let me participate and observe their discussions with other team members. These discussions were generally on topics such as providing an introduction to a new project, or involved some crucial changes within the project, or related to the employees in the company in various ways. Apart from attending these meetings, I also managed to conduct a number of formal interviews with some team members and team leaders (see interview section).

3.3.6. Participating in Other Peoples' Lives is Not without Challenges

My impressions of these company environments and about their people is the result of close interaction with the people both during their business hours and their private lives. During this research experience, I spent about fifty hours per week in the companies. Most of my evenings were spent out with the company workers. It is normal in Jordan for colleagues to go out to socialise after work and spend some time relaxing together. As I have mentioned above, most of the colleagues were friends or relatives anyway, and they liked discussing their family matters over a cup of coffee after work. After being accepted as part of the organisation, I was invited to spend evenings out together with other colleagues, and later was invited to attend two wedding parties and play soccer games, which were a weekly activity. Enlarging the circle of trust I had among my colleagues allowed me to see many things that I would not have noticed otherwise. I was accepted by others and I felt like 'one of them'. It also gave this study its depth and strength. Being so close to many of my colleagues allowed me to learn about their lives in more detail, experience their dilemmas, frustrations, routines, and share happy moments. I felt like a close friend or a family member in that working environment.

Even though the close and friendly relationship I eventually had with the staff provided rich data for this research, the situation I found myself in did not come without challenges. Right at the very beginning of my presence in the companies there was a short period of time when I found myself a stranger in each company. First of all I was overwhelmed by all the new faces and activities happening every day. I found myself exhausted at the end of each working day because I was focusing so much on each detail, being afraid of missing something, and worried about embarrassing myself by asking extra questions. Once I was more involved in the everyday life of the companies and more aware of their usual business process, I was not afraid to miss out on anything or to ask any questions. Also, as relationships were established with the other staff members my work environment became easier for me.

Figure 3.3.6-1 Analysts as friends

The challenges did not end, though, with my establishing good relationships with the other company members. During the later stages of my research, I found myself in a situation where my role in the company shifted. Since I was conducting a thorough investigation of the business process in the organisations, people started seeing me in a new light. My status as a researcher changed to being an ‘expert and a messenger’. When I originally arrived at these organisations, I had the intention of completing my research, collecting some data, and staying in the background as an observer, especially as I had arrangements with the managers that I was to conduct research without interrupting the companies’ life or initiating any changes. All of my colleagues soon came to know that I was a PhD student. Over time, however, since the companies’ environment made it possible and since I wanted to investigate deeper into the companies' lives, I became friendly with my colleagues and learnt more things about their personal lives. Since I was open to any communication and I conducted interviews on a regular basis, people started asking more questions about my past and about my present role in the company. To remain impartial and in order not to undermine my research findings or, even worse, change the company members' points of view about specific information I was interested in, I tried not to reveal the main focus points of my research. However, in due course, people started noticing that I had direct communication with both managers and analysts on a daily basis. This gave them the idea that I might act as a mediator for any issues that they would otherwise be afraid to voice themselves. Staff members started seeing me as an expert who knew both sides of the coin: the management and the analysis. So, for the analysts, I became a mediator who was able to bridge hierarchies and communicate issues they were facing, in order to try to ask for some managerial changes. For the management staff, I became an 'external consultant' who could observe and advise on what was happening in the analysts' working environment and report the level of acceptance of any managerial changes implemented.

This shift in my role developed gradually and it was too late for me to reverse the changes when I realised what was happening. When I understood my new 'position' I did not feel very comfortable. My intention had been to stay in the background throughout my research project, collecting data, and do my internship tasks as initially discussed with the managers. Never did I imagine finding myself doing 'action research', consulting, or negotiation work.

I realised that both analysts and managers were asking me an increasing number of questions related to the everyday process in the organisations. Although I tried to help the staff members out with their internal issues, I never forgot the primary purpose behind my coming into the organisations. With the sudden new workload that was 'assigned' to me, it became hard to balance my own responsibilities and the 'new role' tasks. But cutting down on the new responsibilities might offend the staff and might backfire by creating a negative result for my PhD work; meanwhile, in order to complete my research, I did need to remain within the organisations.

This discussion of the research clearly indicates the practical impact that can follow from choosing particular research methods. The method of data collection I chose directly affected the outcome of my research project. I would not have been able to describe any of the work settings or work environment in the organisations if I had sent out a questionnaire or carried out impartial interviews and observations without actually taking part in the whole work process. However, this study approach left me no choice but to become included in the work process and the organisations' lives, which, in turn, revealed the vivid atmosphere of the companies (Yin 1994). At the same time, this particular approach caused challenges, associated with making sure I did not lose out on details, and with maintaining the role of researcher throughout the whole research project and not being side-tracked by other tasks or expectations. The most challenging problem for me was to keep the trust I received from employees in the companies as a colleague and researcher, but most importantly, as a friend.

3.3.7. Enacting the Methodology.

In this section I shall explain the challenges I faced while conducting ethnographic research in the context of the Jordanian small software enterprises. Because of the very similar context surrounding each of the two organisations involved and the very similar behaviour exhibited by the participants in each organisation, my explanatory discussion of the challenges faced during my ethnographic research will consider both organisations simultaneously.

I chose to use an ethnographic research method because it allows the researcher to acquire in-depth understanding of phenomena and to come to understand phenomena from the practitioner's perspective. This is possible because the researcher can engage in the life experience of practitioners in their day-to-day practices, rather than in just reflective interviews.

From December 2011/ January 2012 onward, the first task I engaged in was to study and come to understand the context of the small software enterprise setting so that I would be able to identify what events and activities small software enterprises are involved in, in terms of PS implementation. I acquired this knowledge by means of conducting interviews with participants from within the organisations, and by participant observation. The results garnered from this data collection phase were used as the basis of an initial code template in order to use as a base for further data collection and analysis. Appendix B in the thesis provides the code template and the outcome of comparisons that were made of how often the codes were inducted from data collected from the two organisations. It also provides information about the process that was involved in the packaged software implementation.

However, I faced a challenge in terms of restrictions being placed on the types of questions I could ask within the organisations and the types of data I could discuss. The participants within the organizations and the top management of these organisations did not want me to discuss sensitive data in my research from their perspective: these included such things as internal company issues, product issues and plans, and issues to do with specific analyst-client relations. There were also some cultural sensitivity issues related to the Jordanian context. I therefore faced a major difficulty when attempting to discuss some issues with participants. Such reluctance on the part of staff in the organisations to discuss sensitive or controversial issues is relatively common in an Arab culture. For example, the website "Arab Business Etiquette" states that:

> *"One of the most important is not to talk about anything that is even remotely controversial. While in the west we talk about politics all the time it is not done in the Arab world and it could put your partners in an embarrassing position. Don't put them in a spot where they need to criticize somebody" (Arab Business Etiquette, 2013)*

One particular occurrence when I conducted observations in Organisation One made me realise the degree to which the analysts in Organisation One did not have a strong relationship with a particular client. The occurrence in question happened when analysts from Organisation One asked me to participate in their discussion with one of the users of inventory software because the user was not collaborating effectively with analysts during the identification of the misalignments process. I accepted the task and we prepared for the meeting with the users. The first action I took during such preparation was to ask for details about the problem with the user. One analyst answered:

> *This user never accepts what we do for him. This project is big and we have been told to do whatever users want, so we did. But this user keeps changing his mind and he keeps asking for new things, so we do them. But at the next meeting he asked us to add to the software function - so why didn't he tell us everything he needed from the same function at once?*

Another analyst explained:

> *Once we got his needs we did them, but he keeps things hidden, so at the next meeting he added some things. But we had already customized the software based on the first needs he had mentioned. I think he was playing with us.*

It was clear from the analysts' comments that they did not trust the user. It also appeared that the problems they had with the user were related to a lack of communication, and that the analysts tended to have argumentative discussions with the user rather than discussions aimed at understanding. I asked the analysts to print out all the software functions that we were going to discuss with the user and the software screen that related to those functions. We arranged a meeting with the user and went to his organisation.

One thing that impressed upon me right away was that the analysts and users looked at each other in a worried and apprehensive way. Since I was from the same culture as them, I could tell that they were not welcoming each other. It was clear that a friendly relation had once been there but that they had lost some respect for each other. This was understandable since the project had run for one year and half and they had had many argumentative discussions. The analysts introduced me as a consultant who would try to help both parties involved to achieve a point of understanding. First, I introduced myself and noted that I was not working officially with the analysts, so I was a third party. I started by mentioning why we were meeting and what we would try to achieve. I tried to keep smiling and to steer the discussion past any previous arguments between the user and the analysts. During the meeting, I acted as a facilitator, trying to keep the meeting focused on its purpose. I found it interesting that I was welcomed very warmly by the user. The user invited me for drink during the break, and one analyst later said “he never invited us to anything before but now he is trying to act nice”.

Another surprising thing was that the user was open to me and discussed all of his needs, even adding everything he needed to have done on the print software screen, and signing it. Directly after the meeting, I discussed this with the analysts. I mentioned that the user was open and had good knowledge, but one analyst replied to me: “you will see, he will change his mind tomorrow”. However, the result was actually that after one month, the user signed the acceptance letter for the analysts to develop the software. It therefore appeared that there was not necessarily anything particularly “difficult” about this particular client at all, but that in this case, interactions between the software development company and their client had been soured by lack of trust. From this, I conclude that discussions based on arguments, or straying away from the meeting’s purpose (engaging in recriminations instead of identifying needed functions and solutions) will harm social interactions and good feelings between users and analysts. On the other hand, in some cases, using a facilitator or a third party as a consultant may help analysts and users to achieve good social interactions.

However, it proved impossible for me to gain any further information about what exactly had gone wrong between them and this client and how their relationship had been managed up until this point. This was impossible because when I attempted to discuss the case further with the analysts and to ask how the relationship with the client had reached such a bad point, the analysts looked offended, as though they thought I was criticising them. Therefore, I stopped asking them such questions. Again, this need to avoid giving offence is common and ingrained in Arab culture, as mentioned in the following quote:

> *“It is generally not considered acceptable to criticize somebody in public in the Arab world so not only do you want to make sure that you*

> *don't put somebody in a position where they may have to but you have to make sure that you don't criticize anybody. This includes disagreeing with them in public; this can cause great embarrassment and offence. Saving face is an important aspect of Arab culture so you have to make sure that you don't cause somebody to lose face by criticizing them. This can be difficult at times but it is something that you have to do" (Arab Business Etiquette, 2013).*

Such desires to save face and to avoid criticism may be behind some of the occurrences I experienced, such as a general manager from Organisation One telling me that:

> *You are welcome to collect data for your study about our practices for software implementation but we would like you to avoid any discussion related to our issues with clients or our internal policies.*

A general manager from Organisation Two told me a very similar thing:

> *Jordan is a small country and we all know each other, all business in Amman - so it will be great if you don't discuss our product issues within your study as well as our internal issues and polices. You are part of our company now so this data is confidential to our company only.*

Due to such limits on what data I could collect and what issues I could ask about, I focused on identifying and understanding the process of packaged software implementation and the practices analysts use during packaged software implementation in order to identify users' needs. I avoided chasing after or discussing any information that was culturally sensitive. Due to such restrictions, I was not able to obtain information about such concerns as software coding issues, product plans, and product release plans. The imposition of such restrictions also meant that the main research question of my thesis had to change from "how is requirements engineering enacted by small software enterprises in the packaged software context", to "what practices do analysts use in order to identify users' needs in the context of packaged software implementation by small packaged software vendors in Jordan?"

Appendix C in the thesis provides an overview of what I had discovered about the process involved in packaged software implementation and about some of the practices conducted by analysts by the time I had progressed through one and a half months of my ethnographic study. Again, the overview of what was learnt during this time avoids revealing any sensitive data.

I understand that ethnographic research "is a form of research focusing on the sociology of meaning through field observation. The goal is to study a community of people to understand how the members of that community make sense of their social interactions (Robinson et al., 2007)". However, Easterbrook et al. (2007) state that for software engineering "ethnography can help to understand how technical communities build a culture of practices and communication strategies that enables them to perform technical work collaboratively. An ethnography might focus on a broad technical community (e.g. java programmers in general), or a small, closely knit community (e.g. a single development team)". Therefore, this study has the aim of understanding the culture of analysts' practices in the context of packaged software implementation.

Easterbrook et al. (2007) identify the preconditions for ethnographic study as including: "a research question that focuses on the cultural practices of a particular community, and access to members of that community". Easterbrook et al. (2007) also suggest that the precise boundaries of the community may not be known in advance. Moreover, ethnographic research adopts a constructivist stance; that is, it takes note of how members of a community construct their own social and cultural practices and structures, and how they view themselves as being defined by these practices and structures (Easterbrook et al., 2007).

Because of issues related to cultural sensitivity and confidential data encountered within Jordanian context, and the limited scope of this study, I changed the data analysis approach I was using during the first one and half months. I began to follow the analysis approach of qualitative data adapted from (Wolcott, 1994; Sandelowski, 2000) as a day by day descriptive analysis of analysts' practices in the context of packaged software implementation (see Appendix D in the thesis and Appendix A in this report). The following quote from Wolcott (1994) Agar describes a process very similar to that I followed: "In ethnography … you learn something ("collect some data"), then you try to make sense out of it ("analysis"), then you go back and see if the interpretation makes sense in light of new experience (“collect more data”), then you refine your interpretation (“more analysis”), and so on. The process is dialectic, not linear” (Agar, 1980, p. 9, as cited from Wolcott, 1994).

This study approach involved process research and aimed to understand the sequence of events leading to a result over time “practice-oriented research”. Rowland (2005) realised that in order to get at the ‘how’ and ‘why’ of decision-making behaviour he needed to conceptualise “the problem of training participation as a process of socio-technical innovation”. He began to put forward a case that on-the-job training schemes were a new innovation in how firms acquire skills. Such a viewpoint departs from previous investigations of on-the-job training that tend to see this development as driven by economics. He also decided to engage in “process-oriented” research rather than “variance” research in order to examine organisational phenomena, and adopted, in particular, the process model of Second Generation Process Theory, which has the objective of reaching a better understanding of how and why participations conducted these processes and practices. Rowlands then argues that it was these ““Who”, “Where”, “When”” elements that “set the boundaries of generalizability” and established “the range of theory (Whetten, 1989)”.

3.3.8. Summary of the Ethnographic Method

During field work, I took detailed descriptive field notes (Appendix D), gathered from the perspectives of different participants. Ethnographic field notes can include observations, impressions, feelings, insights, and emerging questions (Hammersley & Atkinson, 2007). If interviews are conducted, it is very important to record the interviews as the researcher’s memory can never be a reliable source for citation. Quoting from the participants is very important in order to understand their experiences and views. Throughout the fieldwork, I built rapport and trust with the participants, showing an understanding of the practice of their organisations, gaining their respect, and encouraging their wish to participate and help with the collection of data by sharing their own experiences. The most important thing was to show an understanding of the way they worked and to make it clear that I was not pressuring them to adopt any practices or business methods or communication methods other than their own. I realised that it was in people's nature to please the interviewer by saying what they thought the researcher wanted to hear, or what might suit the researcher's hypothesis. In order to mitigate such tendencies, I tried to put interviewees at ease and to encourage them to be frank and uninhibited in their discussions with me. The main focus for me was to collect the data in as raw a state as possible. From the perspective of ethics, I also had to make sure that the participants were not upset, offended, or exploited in any way. It is vital to stay alert during the more mundane part of the research (for example, when information is being processed or analysed) and to stay focused on pulling the data together at the end of the study. Meanwhile, due to the large amount of collected data, I developed my own system for summarising, indexing, and classifying the data as appropriate (see Chapter 4).

In the final stages of my ethnographic research, I clearly separated description and interpretation of the collected data. The analysis process started with the collection of all the raw data and with forming an overview of the entire process. The raw data was then ordered into patterns, categories, and basic descriptive units (see Chapter 4).

3.4. Participants

The population of this research is the Small to Medium sized Software Development Companies (SPSVs) in Jordan, a country in the Middle East. The list of the SPSVs that are developing packaged software was obtained from various software providers in Jordan, who supply both local and worldwide markets. Statistics obtained from the Jordan Companies Control Department reveal that there are 242 SPSVs in Jordan (as shown in the top two rows of Table 3.4-1). As can be seen from Table 3.4-1, there are 190 software development companies in Jordan that the Jordan Companies Control Department regards as 'small' (indicated in the top row of the table), while the number of small and medium companies is 242 (190 plus 52). These small and medium-sized software development companies comprise around 96% of all software development companies in Jordan.

Table 3.4-1 SPSVs based on capital

Capital from	Capital to	Number of Organisations
0 JD	100,000 JD	190
100,001 JD	500,000 JD	52
500,001 JD	1000,000 JD	12
1000,001 JD	10,000,000 JD	6
10,000,001 JD	50,000,000 JD	1
Total	80,765,883 JD	261

(Jordan Companies Control Department, 2010)

Table 3.4-2 demonstrates the recent growth of the software development market in Jordan. Software development companies in Jordan have been progressively growing in terms of the cross-value they add to the economy. The software development cross value added was 54,839 (JD) million in the year 2010. Table 3.4-2 shows stable economic growth from the years 2006 to 2010.

Table 3.4-2 SPSVs by economic activity (JDs 000)

Company type	Year	Cross output	Cross value added	Number of Companies
Software Development	2006	31,884	26,775	190
	2007	33,654	28,566	
	2008	36,443	30,662	
	2009	44,748	34,719	
	2010	46,093	34,901	
Software Consultancy and supply	2006	9,477	6,628	52
	2007	10,344	7,121	
	2008	12,421	8,443	
	2009	22,948	15,170	
	2010	28,654	19,938	

(Jordan Department of Statistics, 2010)

Two software development companies have participated in this research, Organisation 1 and Organisation 2. Organisation 1 was established in 1997. The company has two branches in the Middle East: in Jordan and Saudi Arabia. The branch concentrated on in this study is the Jordan branch, which has 40 employees, including people working in marketing and sales, analysts, developers, and management teams. The services they offer include software development, systems integration, and software localisation. The company's software products deal with accounting, inventories, purchasing, retail, school management, freight management, and human resource management. The company's solution options are categorised into 3 different levels: 1. Basic solutions that are used by small companies who do not have many requirements; 2. Specific solutions, used by companies of various sizes and kinds, and built depending on clients' needs; and 3. ERP solutions used by medium and large size companies, customised to fit with the client's business and needs.

Organisation 2 is a professional IT services company established in 1998. The company, which has approximately 20 employees, is dedicated to providing solutions for the business sector in the areas of agricultural industries and corporations, finance and banking, education, insurance, and health. Organisation 2 deals with four main aspects of information technology: 1. IT hardware supply; 2. data communication installations; 3. IT services; and 4. IT solutions. The company is experienced in developing software solutions (e.g. human resource management systems, accounting systems), mobile applications (e.g. asset management systems, VAN sales, warehouse management systems), and logistics technology solutions (e.g. UPS road net solution, courier management system turnkey barcode solution).

Table 3.4-3 Background of the participants

Position	**Organisation 1**	**Organisation 2**
Team Leader	5(45%)	2(28%)
Software engineer	4(36%)	3(42%)
System analyst	8(72%)	6(85%)
Programmer	11(100%)	7(100%)
Experience (Years)		
Less than 3 years	2(25%)	1(14.2%)
3 to 10 years	8(75%)	5(71%)
11 to 20 years	1(10%)	1(14%)
Experience (Position)		
Analyst	1(11%)	1(14%)
Designers		
Developer	3(33%)	2(28%)
Analyst and Designer		
Designer and Developer		
Analyst, Designer and Developer	9(81%)	4(56%)
Types of Software		
Business Application	8(75%)	5(72%)
Database System	5(45%)	4(56%)

Table 3.4-3 shows the background of the participants who were observed and interviewed in this study. The total number of participants was 11 in Organisation 1 and 7 in Organisation 2. The participants included programmers, analysts, and developers. The team leaders were made up of 5 programmers in Organisation 1 (45% of all the participants from this organisation) and 2 programmers in Organisation 2 (28.4%). Most of the programmers were also system analysts: 8 in Organisation 1 (75%) and 6 in Organisation 2 (85%). The majority of the programmers in both organisations (71%-75%) had a total experience of 3-10 years in the field. However, only 11%-14% of the programmers in both organisations had over 11 years' experience in the field. Likewise, only a small number had less than 3 years' experience.

Most of the participants had experience working as analysts, designers, and developers at the same time. The percentage who had this range of experience was, however, higher in Organisation 1 (81%), than in Organisation 2 (56.8%). Some participants had experience as analysts only, and some as developers only. Most participants had experience with business application software (75% of those working in Organisation 1, and 72% of those working in Organisation 2), and about half of the participants had familiarity with database system software (45% of the participants in Organisation 1 and 56% of the participants in Organisation 2).

3.5. Data Collection Methods Chosen

Ethnographic research was conducted over the course of 7 months from December 2011 to June 2012. Data was collected throughout the research during field work. The three data collection methods, namely, interviews, participant observation, and focus groups, were used due to their suitability for qualitative research.

Myers & Avison (2002) found that many qualitative researchers prefer the term empirical materials to the word data since most qualitative data is non-numeric. In other words, data in qualitative research comes in the form of words, phrases, sentences and narrations which can provide a more complete portrayal of the subject under study than numbers can. Miles & Huberman (1994) describes words and similar data as being capable of providing the rich, full and real story, rather than the thin abstraction produced by numbers. Indeed, these are the data considered appropriate in explaining human and social aspects which cannot be quantified in a universal manner (Myers & Avison, 2002). According to Miles & Huberman (1994), the strength of qualitative data is that it is rich and holistic with strong potential for revealing complexity nested in a real context. The following sections provide descriptions for the three methods for data collection and give justifications for the chosen data collection methods.

Table 3.5-1 indicates the period of time from December 2011 to June 2012 when the study was conducted. Table 3.5-1 states the number of one – on – one interview (I), focus groups (FG) and participant observations (PO) conducted during the period of the study.

Table 3.5-1 Period of time

Month	Interviews (I)	Focus Groups (FG)	Participant Observation (PO)
December	6	0	0
January	4	2	5
February	15	5	5
March	6	6	8
April	12	2	6
May	18	6	8
June	3	2	3

The most often used methods were interviews and participant observations. At the beginning of the research in December 2011, the most utilised method for collecting data was interview. The reason that interviews were the primary method of data collection used at this time is that focus groups take more time to arrange, while participant observations require the researcher to accrue certain time in the field in order to understand the way the organisation works and identify the main participants. In January I concentrated on organising focus groups and focusing on participant observation, and the number of those data collection methods increased. The number of interviews meanwhile dropped. In February the data collection sped up and rose to nearly its maximum for interviews, focus groups and participant observation. In March 2012 focus groups and participant observations continued to climb.

3.6. Participant Observation

Participant observation is a qualitative method that is commonly used in ethnographic research (Myers, 1999). The objective of such observation is to gain multiple perspectives of the population being studied and the relationships between different views held within one community. The data is collected during the field work, when the researcher makes careful detailed field notes of the events, and of informal conversations and interactions with the participants (Walsham, 2006). Taking accurate notes is an important part of participant observation, not only because it records participants' subjective reports about their actions and opinions, but also because it helps to reduce the researcher's bias. Most field notes will consist of textual data; however, the researcher can also note down such things as figures, maps, and charts. Although a large majority of the data will be qualitative, it might also include some of the quantitative information related to the number of participants involved in a particular activity during a specific project. For example, after the original data collection conducted during my ethnographic field work, I engaged in validation of the results obtained; quantitative information related to the validation is provided in Chapter 6.

Participant observation is useful in identifying the physical, social, cultural, and economic contexts of the members of the study, interdependency between people, various ideas and norms held by the members of the study, as well as people's behaviours (Myers, 1999). This method can assist the researcher in understanding the factors that will resolve the research problem, provided that the researcher designs the right questions to ask the participants in order to understand the phenomenon under study (Harvey & Myers, 1995).

After gaining access to the users and analysts' meetings and having my presence accepted by the participants, I began the field work. This allowed me to observe the way analysts interact with the users to collect their requirements. During participant observation notes of the discussions were taken. In cases when the users agreed to be recorded, the meetings were audio recorded. The data collected during these meetings allowed me to prepare appropriate questions for further interviews and focus group discussions, which would reveal the participants' opinions and beliefs about the interactions with the users.

The data collected during participant observation also provided the context for understanding the data. In this case participant observation was done prior to other data collection methods; however, it can also be carried out simultaneously with other methods or even during the data analysis stage.

Participant observation was the main source of data collection. As mentioned earlier, I attended 35 meetings. The type of meeting depended on the project of the software development. The products for development were Human Resource (HR) software, Enterprise Resource Planning (ERP) software, Special Solution software, such as a school management system, Restaurant Management (H2O) software, and Point of Sale (PoS) software. Figure 3.6-1 below visually represents the number of meetings throughout the 5 projects I was involved with.

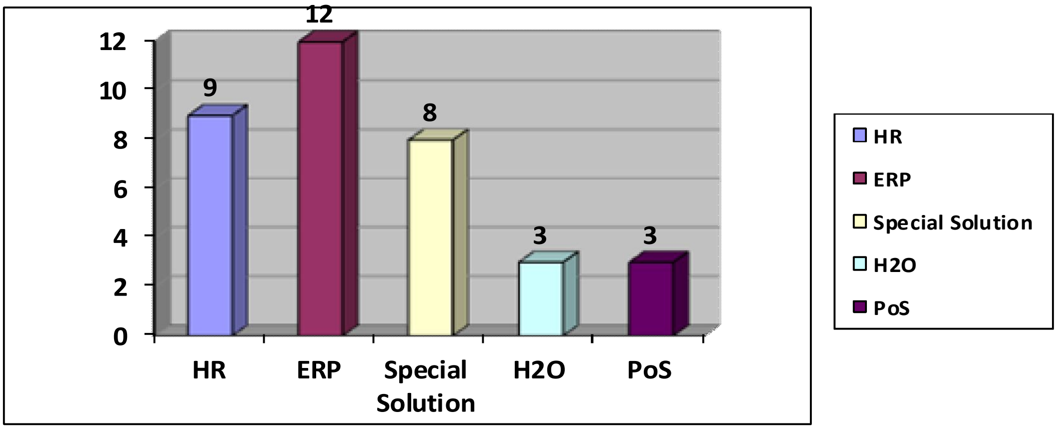


Figure 3.6-1 Number of meetings

Through participant observation, I was involved in 5 different projects: HR, ERP, Special Solution, H2O, and PoS. The project with the highest number of meetings was the ERP project, with 12 meetings.

3.7. Focus Group

In the last decade focus group interviews have become a commonly used technique to collect qualitative data by asking the participants about their perceptions, beliefs, opinions, or attitudes regarding a concept, idea, service, or product (Krueger & Casey, 2008). While social research typically adopts direct observation, focus groups are more appropriate for studies of attitudes and experiences. The communication between participants in the focus group allows the researcher to gain access to various areas for studies and raises unexpected issues for exploration (Krueger & Casey, 2008). Focus groups are used as a self-contained method as well as in addition to other research methods, like in-depth interviews (Krueger & Casey, 2008).

Although there are some disadvantages to conducting an interview in a group, the advantages can be maximised through attention to research design issues and the project and group level. Focus group interviews may encourage people to be involved in the discussion even though they might feel reluctant or unwilling to be interviewed one-on-one (Stewart et al., 2006).

It is of vital importance for the research that these focus group discussions support natural communication by including day to day interactions such as joking, teasing, arguing, and so on. These help to introduce more data, including data that might otherwise have been untapped (Krueger & Casey, 2008). Furthermore, usage of colloquial language allows the participants to cascade and link ideas, and thus extends the preceding topic. When data is collected in such a friendly atmosphere, the researcher analysing the data is better able to pinpoint the shared common knowledge (Stewart et al., 2006).

One of the disadvantages of group interviews involves the social desirability bias - the fact that some participants may remain silent, or that minority opinions might be misheard, or drowned out by the majority opinion (Stewart et al., 2006). The discussions of the employees regarding management might be interrupted and they might be prevented from expressing criticism, even though their discussions are highlighting certain aspects of people's experience. Such dynamics may limit the usefulness of the data for certain purposes since the method does not allow for anonymity (Krueger & Casey, 2008).

On the other hand, group interviews may facilitate the discussion of taboo topics, reveal the opinions common to the group, and encourage people whose experiences are similar to discuss these experiences, when they might have been unwilling to discuss them otherwise. Some research has shown that group discussions generate more criticism than individual interviews (Krueger & Casey, 2008).

Before conducting a focus group, I identified the major objectives of the meeting and developed the main questions relevant to the initial results of the data analysis and research questions. The discussion session normally lasted an hour or an hour and a half, during which I generally addressed 5-6 questions at most. To plan the session, I needed to schedule a time when all the participants could attend. The organisations' conference rooms were used, which allowed all of the participants to see each other. The main ground rules I followed during a focus group were to remain focused on the research topic while the discussion flowed and evolved, to maintain momentum, and to achieve closure of questions. The focus group meeting agendas always included: welcoming the participants, reviewing the agenda and goals of the meeting, explaining the means of recording the session, introduction, conducting a questions and answers period, and wrapping up. During the main part of the session, I might sit back and listen to the discussion. Later on, I might encourage the participants to conduct a discussion in more of a debate style, and to encourage different opinions to be voiced. I worked to facilitate equal participation of all the members, giving each person time to answer the question and to voice their opinion on the matter. To avoid having one or two people dominate the discussion, a round-table rule should be introduced as one of the ground rules at the beginning of a session.

Data collection through focus groups was conducted on Saturdays and Thursdays in both companies. Focus Group discussions held in Organisation 1 and Organisation 2 companies are presented in the Table 3.7-1. I had 23 such meetings with analysts. Some of these were audio recorded and some were not. The main point of the focus groups was to discuss the analysts' perspective about the RE practices in PSI. The analysts also discussed the users' understanding, behaviour, and interaction strategies during the PSI. Some of these focus groups were conducted to discuss the initial findings for this study in order to clarify the analysis result of the study.

Table 3.7-1 Focus group topics

Focus Groups Topics	Questions
Requirements Engineering	What are requirements? How do you collect requirements?
PSI scope, PSI offer, and PS demonstration	What is PSI Scope? How do you define PSI Scope? What are PSI Scope elements? Why these elements are important? What is PSI Offer? How do you create PSI Offer? What are PSI Offer elements? Why these elements are important?
PSI, PS demonstration, PS customisation.	What is PSI? How do you conduct PSI? What is PS Demonstration? How do you prepare PS Demonstration? Why do you demonstrate PS?
PS customisation	What is PS Customisation? How do you decide to customise the software? How do you define customisation needs? How do you minimise customisation?
PS customisation, Identify misfits	What are misfits? How do you define misfits? What are benefits of misfits?

3.8. Interviews

Interviews of individuals are a widely used tool to access people’s experiences and their inner perceptions, attitudes, and feelings of reality (Denzin & Lincoln, 2005). There are three types of interviews: structured interviews, semi-structured interviews, and unstructured interviews (Denzin & Lincoln, 2005). In this study, I chose to utilise unstructured interviews as one of the data collection methods.

The choice of using unstructured interviews as a data collection method was based on my research objectives and based on my adoption of an epistemological stance. My reasons for selecting the unstructured interview are in alignment with those mentioned by Denzin (2000) who states that unstructured interviews help to make sense of a subject’s world through the subject’s own perspective and in their own terms.

The methodology of using unstructured interviews was developed in anthropology and sociology in order to obtain insight into people’s vision of reality. An unstructured interview does not have a set of questions or answers. Different authors use different terms to refer to unstructured interviews, such as “informal conversational interview”, “in-depth interview”, “nonstandardised interview”, and “ethnographic interview” (Denzin & Lincoln, 2005). The definitions of such interviews and their purpose are also numerous. Denzin (2000) described them as a way to understand people’s complex behaviour without categorising it beforehand, thus enlarging the field of enquiry. Regardless of such various names and definitions, unstructured interviews are a best fit for this study. Researchers doing unstructured interviews are open to changing realities and have an interpretive approach to events (Denzin, 2000; Spradley, 1979). They approach the reality from participants’ perspective, interpreting it in participants’ terms (Denzin, 2000).

Unstructured interviews are entirely informal, and rather than being based on a set of questions, they are based on a set of issues that the researcher is interested in and would like to highlight during the interview (Hammersley & Atkinson, 2007). This allows for greater freedom in the exploration of topics, but the researcher’s relative lack of control over the conversation may encourage the interviewees to relate experiences that are not relevant to the problem investigated in the research (Hammersley & Atkinson, 2007; Denzin, 2000). The researcher generally tries to keep his or her control to a minimum by asking spontaneous questions that will be generated based on the interviewee’s narration and the researcher’s reflection on it (Denzin & Lincoln, 2005). The danger of entering into multiple or irrelevant discussions can be controlled by an agenda – a broad guide to issues created by the researcher beforehand (but not disclosed to interviewees), which will create a degree of consistency across the whole interview, helping to achieve a balance between flexibility and consistency.

My choice to use unstructured interviews is supported by Denzin (2000), who argues that they help to collect detailed data regarding people’s perspectives on information and its usage. They are useful for finding specific patterns, which define models. For example, Alvarez (2002) in his study used unstructured interviews to examine information requirements during the implementation of an ERP. In my study, this approach helped me to note down all the topic areas covered during each interview. Unstructured interviews also proved beneficial in my study by revealing some related areas, which I had been unaware of. They also helped me to avoid bringing my own pre-framed or possibly biased understanding of RE into the data collection. Ultimately, they generated detailed data which provided in-depth understanding of a phenomenon.

3.9. Utilisation of Interviews

Analysts were interviewed according to the schedule discussed and agreed on with the participants as well as at times that had not been previously arranged. In fact, skilled ethnographers often gather most of their data through participant observation and many casual, friendly conversations. They may interview people without their awareness, merely carrying on a friendly conversation while introducing a few ethnographic questions (Spradley, 1979).

Table 3.9-1 shows the interview times with analysts. For the Special Solution project I had interviews with 4 different analysts. For ERP I had interviews with Participant 1, Participant 2 and Participant 3. Participant 4 and Participant 6 were interviewed for HR. Participant 6 had another interview for PoS. The interviewee for H2O project was Participant 7. In total I held 71 Hours of interviews with different analysts for the 5 projects that I was a part of.

Table 3.9-1 Interview times with analysts

Analyst	**Project**	**Time**
Participant 1	ERP	13 Hrs
Participant 2	ERP, Special Solution	8 Hrs
Participant 3	ERP, Special Solution	4 Hrs
Participant 4	Special Solution	4 Hrs
Participant 5	Special Solution	11 Hrs
Participant 6	HR, PoS	14 Hrs
Participant 7	H2O	6 Hrs
Participant 8	HR	11 Hrs
TOTAL		71 Hrs

The interviews provided me with background information from the analysts' perspective. All interviews were in conducted in a way that would encourage a relaxed atmosphere in which the analysts would feel comfortable and speak freely. This was achieved by organising the meetings in the participants' spare time at a location convenient to them. The participants also knew the researcher prior to the interview and it was their decision to participate in the study. There were no outsiders present during the interviews. It was confirmed that all the data collection files and documents were to be kept at a secure location without any disclosure to others. To make the participants feel comfortable discussing their projects while being observed, they were allowed to stop the audio recording at any time they felt uncomfortable. In case of embarrassment or any discomfort, the participant might choose not to answer the question asked, or ask to stop audio recording completely. In the case of any psychological distress, the participants could choose to take a break and help themselves to hot beverages and snacks.

Prior to their first interview each analyst was provided with an information sheet, which outlined the idea for my study, the purpose of the research and the way it was going to be conducted. For the information sheet, see Appendix A. The information was also presented orally, to clarify the information and to enlarge on the ideas in the information sheet. In the case of any misunderstanding or lack of information, the analysts' questions were answered and the information was discussed. This discussion also gave me a chance to modify and refine my protocol and put information into a clearer form. The analysts also signed a written consent allowing me to record, transcribe and analyse the interviews for my research, which is important for an interview-oriented study.

The analysts answered the initial questions (not in all cases) that would give me an overall idea and the background information to identify key points for further discussion. Then the analysts and I discussed a list of general topics and issues. During this section of the interviews I did not interrupt the participants, but rather let them express their thoughts about the topic. The interviews were audio recorded, and my note taking process identified the key points for further open discussion with the participants that would bring in a more in-depth focus on the situation. Some of the points divulged by the analysts revealed extra information in the area of interest for me. These areas were then discussed in more detail.

Using unstructured interviews gave me the ability to code and to identify relevant patterns while analysing the collected data. The text for analysis was prepared from audio files and notes taken during the interviews. Most of the audio files were transcribed. I read and re-read the written text (transcriptions of the interviews) from the interviews with each participant to become more aware and familiar with the information provided. This engages the researcher into the meaning of the text and helps them to get an initial interpretation of the text, which facilitates coding. During the coding phase, the data revealed particular patterns, which helped to prepare me for the second interviews with analysts.

After identifying the patterns and preparing questions for the following discussion, I met again with the analysts. The structure of those meetings was driven by the data analysis and results from the previous interviews. The purpose of these meetings was to link my understanding and the participants' perspectives. The researcher's understanding was double checked and confirmed with the analysts by providing feedback to the participants raised in the previous stage and by probing questions during the interview. The central aspect of producing findings from this interview was supported by rich and in-depth discussion of the participants' experience.

Most of these subsequent interviews were also audio recorded for further analysis and coding. During the interviews I took notes, to direct the discussion to the particular points of interest, and get more clarification on the points not explained thoroughly.

All interviews were conducted in the same style: by using a lot of clarifying questions, which allowed the exploration of concerns relevant to the topic. Unstructured interviews allowed me to obtain further information and to probe the themes that I was not initially or not fully aware of. Allowing the participants to freely express their point of view on the topic may minimise interviewer bias.

3.10. Working with Two Organisations.

In this section I will explain the data collection approach I used within both organisations. First of all, the geographic address for both organisations was the same: the two organisations were actually located in the same area and street in Amman. This gave me an opportunity to move between them very easily. Secondly, both organisations were similar in terms of context and in terms of their business process approaches. As I mentioned in Section 3.3.7, during the first one and a half months of my study I was focused on understanding the business processes of both organisations. Based on the understanding that I gained, I provided business process diagrams (see Appendix C) showing my understanding of and interpretations of the organisations' business approaches. Both organisations agreed that the business process diagrams that I created properly reflected their business processes.

Collecting data from two organisations actually assisted me in being able to make adjustments to the ways that I collected my data. Lessons learned from the data collection I carried out in one organisation helped me to adjust data collection in the other organisation, and vice versa. One lesson I learned related to the fact that businesses in Arab cultures have unstructured time management. My involvement with Organisation One taught me that it was best to set up Focus Groups both at the end of the day (as opposed to during the day) and at the end of the week. I therefore began to schedule all Focus Groups to occur on Thursday evenings. The next alternative that the organisations preferred was Saturday morning. These preferences were culturally related: Thursday evening is the end of the working week in Jordan, Friday is a holiday, and Saturday represents the beginning of the next working week. Participants from the organisations preferred to have Focus Groups on Thursday nights or Saturday mornings precisely because the Focus Groups were seen as enjoyable discussions, rather than as a form of work. A Focus Group on Thursday night therefore represented a nice winding down of the working week, whereas a Focus Group on Saturday morning was seen as a relaxing way to be introduced back to work.

The second lesson I learned as I collected data that led to my making adjustments to my data collection methods related to the discussing of RE practices that I had adapted from Sommerville & Sawyer (1997) and Cox et al. (2009). In Organisation Two I found that analysts had a problem applying theoretical terminology to describe their requirements engineering practices. For example, when I provided theoretical names for the practices adapted from Sommerville & Sawyer (1997) and Cox et al. (2009), they could not always identify which practices were being referred to. I therefore adjusted my approach in this regard by providing verbal explanations of the various practices and then providing written definitions and explanations of each practice and its purpose, in Arabic. Providing the analysts with verbal and written definitions in this way clarified their understanding of the concepts and terminology I was referring to and thus allowed them to be much more prepared to contribute to discussion during Focus Groups.

The third lesson that I learned from both organisations was that data collection would proceed more smoothly if I could avoid asking questions that could be perceived as too sensitive: for example, I learned not to ask any analyst what they thought about the work completed by another analyst, as this made them uncomfortable.

Apart from these changes that I made during the course of my involvement with the two organisations, I did not make any major adjustments to the data collection methods I was using. It was not necessary to change many data collection methods as I found that, on the whole, the methods that I used were successful and effective. Therefore, any adjustments that I made during my time with the companies were related to finessing my interactions with the participants, rather than involving full-scale changes of approach.

3.11. Ethical Conduct of the Research

Ethical approval for this research was obtained from the Auckland University of Technology Ethics Committee (07/02/11, #10/293). One of the main concerns related to ethics was getting informed consent and providing the participants with confidence regarding taking part in the research. By informed consent we mean the voluntary agreement of an individual to take part in a research project, based on his/her understanding of the nature and purpose of the research itself (Sim, 1986). Informed consent can have four constituents: disclosure (providing adequate information), comprehension (understanding the information), competence (ability of participants to make a rational decision), and voluntariness (no compulsion) (Sim, 1998).

To avoid any lack of confidence and misunderstanding, all participants were provided with the aims of the research in a written form. The information sheets containing this information were either presented directly during a face-to-face meeting or sent by email. After familiarising themselves with the document, the participants had a chance to ask questions about the research and clarify the objectives and process. All of the participants were made fully aware of the fact that they could withdraw from the research at any time without any consequences to themselves. The participants were engaged only after verbal explanation and after they had voluntarily signed the written consent and information statements prior to the research. The researcher did not use any power relations towards the participants in order to collect the written agreements.

Confidentiality, as one of the main ethical concerns, was maintained by changing real names in the research report and substituting some specific details related to the identity of the participants to keep the personal nature of the research private.

3.12. Analysis Method Selection

The literature regarding qualitative research methods places great emphasis upon the methods used to go out and collect or generate data, but less emphasis upon the analytical techniques that can be used to interpret these data. So whilst different approaches might be taken when conducting qualitative research, there are also requirements that there should be some consistency between methods, methodology and analysis, in order to demonstrate the story being told. At the same time, in order for the research to be credible to the reader, the reader needs to be led toward what the researcher thinks is most significant about the research findings (Gregor, 2006; Denzin, 2000). The importance of these findings must be made "transparent" and choices and assumptions made by the researcher made explicit in relation to the methodological perspective (Klein & Myers, 2001). For example, if a researcher is a positivist and tends to use deductive method reasoning, they'll tend to do "this". If they are an interpretivist and tend to use inductive method reasoning, they'll tend to do "that".

A wide range of literature documents the underlying assumptions and procedures associated with analysing qualitative data, including the evaluation of data and data analysis strategies, and inductive and deductive approaches. Many of these are associated with specific approaches or traditions, such as grounded theory (Strauss & Corbin, 1998), phenomenology (e.g., van Manen, 1990), and thematic analysis (e.g., Boyatzis, 1998).

In this study the inductive approach is used. The inductive approach, in this study, refers to a data driven approach which starts with the close study of collected data (based on day by day descriptive analysis from participants observation, interview data and focus groups) an interpretation of the collected data's possible multiple meanings (Thomas, 2006). The following quote from Agar (1980) describes a process very similar to that I followed: *"In ethnography ... you learn something ("collect some data"), then you try to make sense out of it ("analysis"), then you go back and see if the interpretation makes sense in light of new experience ("collect more data"), then you refine your interpretation ("more analysis"), and so on. The process is dialectic, not linear" (Agar, 1980, p. 9, as cited in Wolcott, 1994).*

3.13. Summary

The attempt to understand the phenomenon of Packaged Software Implementation Requirements Engineering in depth and in detail is likely to require a researcher to adopt a qualitative ethnography format for their study. Such a study is likely to benefit most from the ethnographic format because it best fits with the everyday reality of work. This is because the understanding of users' requirements is marked by complications, struggles, and other specifics that are "part and parcel" of knowledge exploitation. This research represents an initial attempt to understand Requirement

Engineering for Packaged Software Implementation and, consequently, users' needs, influencing skills and sharing attitudes of analysts are understood to be an integral part of the dynamics of Requirement Engineering for Packaged Software Implementation.

Companies' practices when implementing packaged software are not well understood in terms of requirement engineering. This is achieved by understanding the "best practices" in functioning requirement engineering activities that are applied by the software development companies. By and large, collecting qualitative data through the use of an ethnographic approach and analysing such data via inductive methods is the most appropriate methodology for exploring the requirement engineering involved with Packaged Software Implementation.

Chapter 4 Data Analysis Strategy

4.1. Introduction

The previous chapter of this thesis focused on the research approach and data collection methods used throughout this study. This chapter deals with the selection of an appropriate data analysis.

In this chapter I provide a description of the process that I followed to choose a data analysis strategy. The data analysis strategy used depends on the research philosophy of the researcher and the types of data that the researcher collects during their study. Since my research philosophy inherently drove the conduct of this study to follow the interpretive paradigm, and the data is predominantly qualitative, I chose a congruent data analysis strategy. This study thus uses an inductive approach, which allows theory to emerge from the raw data. Therefore, I did not analyse the data I collected based on previously existing findings. This does not mean that a researcher should not have knowledge about previous findings; rather, researchers should avoid letting such knowledge bias their analysis of newly collected data (Allan, 2003).

There are many strategies that can be used to analyse qualitative data inductively, such as thematic analysis, content analysis, and grounded theory. However, the most important issue is how one deals with the data during the analysis phase.

> *"In qualitative research the process by which data analysis is undertaken is fundamental to determining the credibility of the findings. Essentially it involves the transformation of raw data into a final description, narrative, or themes and categories. There is considerable variation in how this is undertaken, depending on the research question and the approach taken. What is important is that the process is described in sufficient detail to enable the reader to judge whether the final outcome is rooted in the data generated."*
> *[Holloway & Wheeler, 2009]*

In this study, qualitative data analysis was undertaken using a three-stage inductive analysis approach (that is, a data driven approach). In this study, inductive analysis is an approach to dealing with data that involves the creation and application of low-level codes, categories, and finally, of themes. The organisation of this chapter is as follows: Section 4.2 provides a general overview of the inductive analysis strategy; Section 4.3 provides details of the conceptual definitions of theory and generalisability, based on an inductive analysis approach; Section 4.4 discusses the evaluation the quality of the data analysis results.

4.2. Inductive Analysis Strategy

An inductive analysis (or data driven) approach starts with the close study of collected data (based on day by day descriptive analysis from participants observation, interview data and focus groups) an interpretation of the collected data's possible multiple meanings (Thomas, 2006). The following quote from Wolcott (1994), Agar describes a process very similar to that I followed: *"In ethnography ... you learn something ("collect some data"), then you try to make sense out of it ("analysis"), then you go back and see if the interpretation makes sense in light of new experience ("collect more data"), then you refine your interpretation ("more analysis"), and so on. The process is dialectic, not linear" (Agar, 1980, p. 9, as cited from Wolcott, 1994).*

The researcher identifies parts of the collected data that are important to the research (see figure 4.2-1) and labels them, assigning them to various categories. After identifying a category and its meaning, the researcher writes a note about the category (for example, its associations, links, and implications) and its links to other categories.

Using a qualitative research method such as inductive analysis brings to light some challenges involved in the data gathering and data analysis processes. The distinction between the gathering of data and the analysis of such data is clearly outlined in theory, but has some difficulties in practice. When using an inductive approach, the researcher gathers the data first and builds his/her theory based on the results of the data analysis. There is a strong belief, however, that the researcher's prior knowledge and suppositions can affect the data collection process via the questions directed to participants (Myers & Avison, 2002; Thomas, 2006). These questions directed to the participants may guide/lead them, and thus may affect or circumscribe the findings. It may be more accurate, therefore, to speak of a "mode of analysis" rather than of "data analysis" in qualitative research (Myers & Avison, 2002).

Inductive analysis can be treated from two perspectives (Fossey et al., 2002). From the philosophical point of view it provides good grounds for interpretivism, while as a mode of analysis, it allows researchers to understand data contained within textual information (Boyatzis, 1998; Thomas, 2006). It is often regarded as a good way of analysing qualitative data as it allows patterns to be found in data via the process of coding, and at the same time supports the interpretation of the data by synthesising parts of the whole (Thomas, 2006). In this work, I have coded the initial raw data inductively.

The study consisted of several stages and cycles. In order to define the events and situations that cause requirements engineering to occur during packaged software implementation I conducted a series of participant observations and interview.

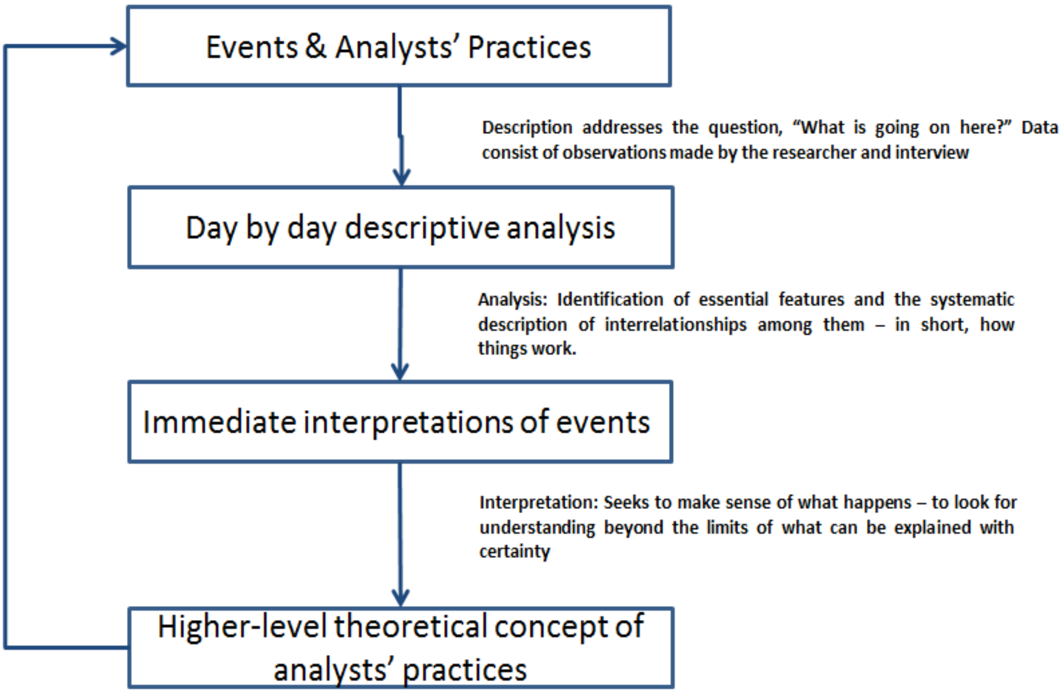

Events & Analysts' Practices
Description addresses the question, "What is going on here?" Data consist of observations made by the researcher and interview
Day by day descriptive analysis
Analysis: Identification of essential features and the systematic description of interrelationships among them – in short, how things work.
Immediate interpretations of events
Interpretation: Seeks to make sense of what happens – to look for understanding beyond the limits of what can be explained with certainty
Higher-level theoretical concept of analysts' practices

Figure 4.2-1 data analysis cycle (Wolcott, 1994)

The inductive analysis procedure that I followed was similar in kind to the types and arrangement of approaches advocated by Wolcott (1994). Wolcott, writing about transforming qualitative data into useful insights, posits that qualitative research involves three major and distinct subsections or phases, within which researchers may take up different approaches to presenting data. He refers to these wider phases as "description", "analysis", and "interpretation". The methodology I followed also divided my qualitative analysis into these three different forms, as indicated in Figure 4.2-1 above. Through field notes detailing events and participant observation and day-by-day descriptive analysis, I fulfilled the "description" form of qualitative inquiry discussed by Wolcott. My immediate interpretations of events and day-by-day descriptive analysis presented facts relating to what I had observed and then sought to identify interrelationships between different events, ideas, and processes involved in PSIRE. This is the form of inquiry described by Wolcott as "analysis". Lastly, I moved on to "interpretation", seeking to make sense of what happened within the software companies studied by providing high-level theoretical readings of events and practices.

While I faced a challenge in terms of restrictions being placed on the types of questions I could ask within the organizations and the types of data I could discuss (see section 3.3.7), I found that following the kinds of analytical approaches for qualitative research advocated by Wolcott (1994), Sandelowski (2000) and Poba-Nzaou & Raymond (2013) was beneficial. After collecting data, I placed particular importance on carrying out a day by day descriptive analysis of analysts' practices in the context of packaged software implementation, and (as demonstrated in the figure 4.2-1 above) as I engaged in analysis and interpretation I went through an iterative process of looking back at my earlier field notes and descriptive analysis.

4.2.1. Events & Participant Observation

During ethnographic research the ethnographer goes through a learning process at the same time as conducting their research. This process allows any prior presumptions that are found to be false to be redefined, reformed, or discarded (Hammersley & Atkinson 1995). After analysing the data I had collected during my field work, I realised that the fact that I did not find what I had expected to find was actually an advantage. My initial expectations of what I would find about requirements engineering were in line with how analysts conduct traditional requirements engineering. However, I remained open to experiencing what was going on around me, to paying attention to the details of the process, and to observing what was actually happening in the companies, rather than trying to search for relevant data. I began to reflect on what actually occupied the analysts rather than on my own ideas or presumptions as a researcher. In my field notes, I was noting down information about everyday activities as well as talks between the employees in the companies. I soon noticed that the unstructured interviews that I had with the participants were revolving around the same issue: the inconsistency of users' needs. Eventually I realised that this was the actual concern of the analysts – issues related to misalignments: "we end up in an infinity loop once you start to implement our software" [General Manager and Team Leaders]. This concern felt by the analysts had a tremendous impact on the day-to-day work carried out within the companies. After realising that there were real concerns regarding the packaged software implementation issues related to misalignments, I started asking the analysts about how the situation used to be in the companies, as well as what their thoughts were regarding any changes. As soon as the participants in my study understood that I was interested in this issue, the analysts started feeling very passionate about the changes and came up to me to express their opinions even without me approaching them to ask them about this issue.

According to Spradley (1979, p.92) "before proceeding to next interview it becomes necessary to analyse the data collected. This is enabling you to discover the questions to ask in future interview" [sic]. Figure 4.2.1-1 below shows the ethnographer (in this case, myself) collaborating with analysts during periods of fieldwork. This collaboration is subsequently referred to throughout this thesis as the situations involved during 'PSIRE'.

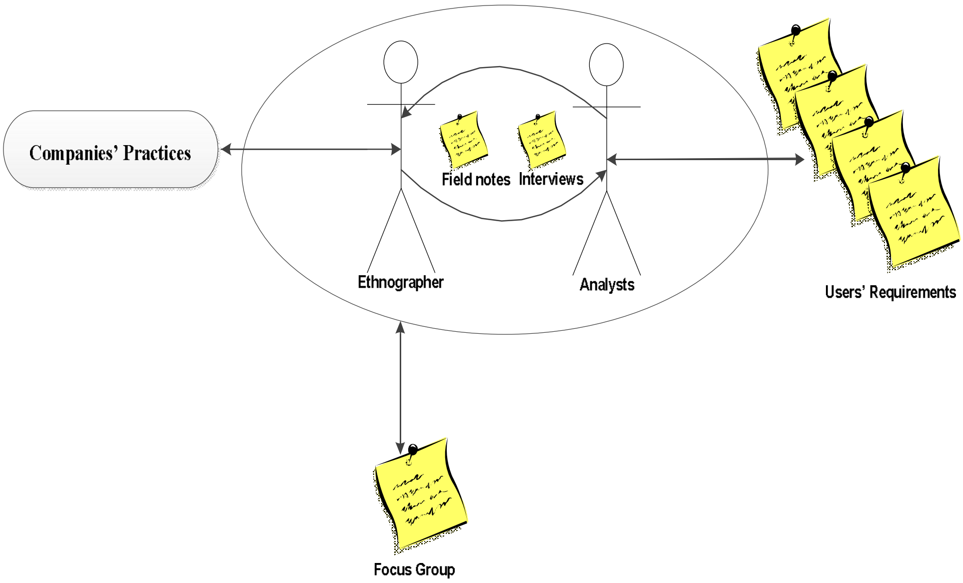


Figure 4.2.1-1 Ethnographer's collaboration with analysts

I also engaged in field observations of events in the companies so that I could discover the situations in which PSI occurs and understand the RE process analysts apply in terms of PSI. Two methods of recording observations were used: note taking and tape recording. According to Fossey et al. (2002), "note taking and tape recording is a useful combination that enables analysis of the material as a whole, while more specific components of interviews can be transcribed in full for detailed analysis" (p. 728). During field observations of events, at the same time as trying to understand what process analysts apply, I built a trust relation with the participants, interacting socially with them during coffee breaks or lunch time. Later I played several soccer games with them to keep up the bond, and I even attended two wedding parties of the participants.

As a result of these field observations, this thesis provides a business process flow chart (see Appendix C) that identifies where PSIRE occurs in organisations. Appendix C represents the organisations' business process from the participants' perspective, and reflects information gained from meetings with every team involved in this process. Circled elements indicate when analysts and users were involved and when PSIRE occurred (see Appendix C). While I had gained a very clear idea of the situations that PSIRE could occur in quite early in my study, in order to get in-depth understanding of the purpose behind the PSIRE applied by analysts during demonstration presentations and after the user accepted a software offer, I requested permission to attend some sessions of demonstration presentations. I also asked permission to attend some meetings (see section 3.6) between analysts and users. These sessions and meetings were tape recorded (whenever possible), and I also took notes. I started observing what actions the analysts carried out during these sessions and meetings. I also asked the analysts' team leaders for interviews since they were responsible for carrying out demonstration presentations.

4.2.2. Day by Day Descriptive Analysis of Analysts' Practices

Wolcott's book on *Transforming Qualitative Data* (1994) answers research questions such as (a) "How do qualitatively oriented researchers transform what they see and hear into intelligible accounts?", (b) "may a researcher present data solely in a descriptive mode, leaving to the reader – or to a future time – the task of analysis/ interpretation?", (c) are the processes of interpretation and analysis virtually interchangeable, or can a distinction be made between them?, and (d) if a distinction can be made between analysis and interpretation, "should researchers try to accomplish both analysis and interpretation in the same study" or direct their efforts exclusively toward one or the other? (p. 1) Wolcott suggests that the real mystery lies not in how the qualitative researcher gathers data but in how they turn the data into an intelligible account of a phenomenon (p. 1). Wolcott uses the more encompassing term "qualitative" analysis rather than the narrower idea of "ethnographic" research (p. 3).

Wolcott (1994) views "description", "analysis", and "interpretation" as the three major dimensions of qualitative study (p. 6), and his work posits definitions and explanations of what he perceives as the distinctions between "description", "analysis", and "interpretation" (p. 5); the suggestions and examples provided in his book are strongly influenced by ethnographic research and by studies by other ethnographically-oriented qualitative researchers (p. 7).

Wolcott (1994) discusses how qualitative researchers can best go about using data: the problem of transforming "unruly experience" into an "authoritative written account" (p. 10). Wolcott suggests that there are three major modes of gathering data: "participant observation (experiencing), interviewing (enquiring), and studying materials prepared by others (examining)". Meanwhile, there are also three alternative approaches to using this data: one can either write an account that stays very close to the data as originally recorded, allowing the data to speak for itself (this approach Wolcott regards as "description"); expand one's written account "beyond a purely descriptive account with an analysis that proceeds in some careful, systematic way to identify key factors and relationships among them" ("analysis"); or engage in "interpretation" which may not be as restricted by observational data as the "analysis" method is, but rather seeks to make sense of what happens – to look for understanding beyond the limits of what can be explained with certainty (pp. 10-11). Wolcott argues that these three categories, "description, analysis, and interpretation" are not mutually exclusive, but that it is useful to distinguish among the three. He suggests that they may be best regarded as "varying emphases that qualitative researchers employ to organize and present data" (p. 11).

While Wolcott realizes that previous qualitative researchers have sometimes used the terms interchangeably, he offers working definitions of each. Therefore, for Wolcott, "Description addresses the question, 'What is going on here?' Data consist of observations made by the researcher and /or reported to the researcher by others. Analysis addresses the identification of essential features and the systematic description of interrelationships among them – in short, how things work. In terms of stated objectives, analysis also may be employed evaluatively to address questions of why a system is not working or how it might be made to work "better". Interpretation addresses processual questions of meanings and contexts: 'How does it all mean?' 'What is to be made of it all?'" (p. 12).

In his discussion of "description" Wolcott suggests that it is a mistake for qualitative researchers to pass on accounts that are "'entirely' in their own words", that preserve every word spoken by interviewees, or that do not attempt to make sense of things observed. Passing on such "raw" data (rather than "cooked" data) suggests that the researcher assumes a reader can look at the data and interpret it themselves in an objective or valid way – leaving it up to the reader to overcome "all the problems that the researcher tried to avoid" (p. 13). Wolcott explains that it is a mistake to suppose that this would present "pure" experience because when collecting data the researcher already chooses to include some data and leave out other data. Moreover, if a huge amount of data is heaped onto readers and they are left to sort through it, readers may lose interest or begin to wonder what the point of the study is (pp. 13-14). However, Wolcott states that "thick description" can be achieved. The key here is that "every decision about the appropriate level of detail returns ultimately to the immediate purposes being addressed and to overriding considerations that map a course between extremes of too-selective reporting or hopeless obfuscation. Every detail considered for inclusion must be subjected to a critical judgment: 'Is this relevant to the account?'" (p. 14). Wolcott warns that if researchers become too concerned with remaining completely objective, they will fall into the trap of treating everything with the same level of detail. Instead, Wolcott states, it is okay if descriptive narratives zoom in and out from wider pictures to smaller detail, and vice versa (pp. 16-17).

Wolcott explains and discusses ten different approaches that could be used by qualitative researchers when organising and presenting descriptions. However, he also explains that "I have used variations and adaptations of these approaches in developing my own narratives, but always in combination, never in the pure forms described below" (p. 17). The first suggestion that Wolcott makes is that qualitative researchers can present their descriptions in (1) "chronological order". "Events always can be related in the order that they occurred, with relevant context introduced as needed" (p. 17). The second suggestion is that (2) researchers can present a description in "researcher or narrator order" (p. 18). There are logical ways to organize a description other than using a chronological sequence. Instead, the researcher could consider the degree to which a narrator's story arrangement contains its own internal logic before they decide to rearrange it. Another possibility is (3) "progressive focusing". With this approach, "the descriptive account may be revealed through a progressive focusing that goes in either direction, slowly zooming from broad context to the particulars of the case, or starting with a close-in view and gradually backing away to include more content" – or, moving in both directions (p. 18). A further option is to describe (4) a "day-in-the-life". "This approach can take a reader to the scene of the action. The day-in-the-life need not be interpreted too literally. Readers might be privy to a real or a fictionalized account, an entire day, or some customary sequence of events. A variation is to take readers along on a reconstruction of the first day of fieldwork" (p. 19). However, the researcher must take enough field notes that they actually can later describe a day-in-the-life. The fifth suggestion made by Wolcott is to describe "critical or key event[s]". This approach circumvents the problem that no researcher can ever tell the full story of events. The researcher can instead choose to "focus on only one or two aspects, creating a story-within-a-story in which the essence (but not the detail) of the whole is revealed or reflected in microcosm" (p. 19). Description can also be presented by arranging it around (6) "plot and characters", as though the researcher is presenting a play. "First, the main characters are introduced. Then the story is put into motion. At that point the researcher may either fade into the wings or assume the role of narrator" (p. 20). The seventh approach is to (7) focus on "groups in interaction". Sometimes it is helpful "to create distinct group identities to emphasize differences important to a case". Such a focus can draw attention to "change agents" (p. 20). Another suggestion is that the researcher can follow (8) "an analytical framework", since adopting an analytical framework while conducting field work may aid them later on when conducting analysis. Moreover, "adopting *any* framework imposes structure on the descriptive account" (p. 20). The difficulty here is making sure that the researcher maintains some scepticism as they collect data, so they do not collect data that *only* supports the preconceived framework. Another useful approach outlined by Walcott is (9), to use the "Rashomon effect". This means that a researcher presents an event through the eyes of several different viewers, recognizing the idea that there is "not one version of any event but as many versions as there are viewers" (p. 21).

It can be seen that many of the approaches that I followed during field work, analysis, and when interpreting data map to the approaches recommended by Wolcott (1994). For example, during my fieldwork I wrote day-by-day descriptions of events, and my inclusion of such descriptions in this thesis has been guided the principle of whether the details in a particular description are relevant to the ethnographic account. Meanwhile, I have utilised a number of different approaches when presenting the descriptions: in general terms, I have roughly followed a chronological order when presenting descriptions of events, but when presenting particular stories or comments supplied by participants I have often maintained the order of story or comment that was given by the participant. I have generally tended to first introduce the broad context of an issue in question, a practice, or an event, and then moved toward presenting particulars. By presenting information taken from my day-by-day descriptive analysis I have also been able to give the reader an idea of a "day-in-the-life" of a software analyst, or even an idea of a "day-in-the-life" of an ethnographic researcher such as myself while in the field. Meanwhile, since I could not tell the full story of every event that occurred in the software organisations, I have focused on describing the most important key events. Those events I perceived as having the biggest effect on whether the software organisations obtained a particular client and succeeded in developing software that suited the client were the meetings held by analysts to create the software offer, demonstrations at which analysts showed the software to the client, and meetings held between analysts and clients during which requirements and needs were elicited or validated. At times I have also used an approach similar to what Wolcott (1994) calls the "Rashomon effect": while my study focuses specifically on the practices of analysts, from time to time I have presented information about an event from the point of view of the analyst and I have sometimes included more than one comment from the analyst team, for example quoting one analyst's view of an event, a meeting with a client, or of an RE practice, while also presenting a comment made by the Team Leader of the analysts. When first presenting information about the software companies, for example, I described how the management and analysts felt about each other and how they did not entirely trust one another. I also presented quotations showing analysts expressing frustration at how clients behaved, and vice versa.

Poba-Nzaou & Raymond (2013) provide further details of how they assigned data elements to different categories and went about generating an initial framework: their “interpretations derived from the reading and rereading of the interview transcripts (Carroll & Swatman, 2000), and discussions among the researchers led to the development of the initial components of the proposed framework and to their further refinement. Subsequently, a narrative approach was used to describe the case in the form of a narrative report (Langley, 1999). The case description was analysed to discern broader patterns across the adoption process, and this led to the generation of the initial framework”. Meanwhile, “as befits the interpretive nature of this study, our analysis involved interpreting the nature of data elements and assigning a category to them, as well as inferring relationships between data elements […], usually grounded in inductive pattern recognition (Miles & Huberman, 1994)”.

Table 4.2.2-1a-b below provides an example of how the descriptive analysis form and discussions of analysts’ practices contained in this thesis contribute toward building a theory for explaining and understanding PSIRE (for further examples, see Appendix D).

Table 4.2.2-1a descriptive analysis form

Feb 1, 2012 - Organization 1- Pre – Implementation/ Software demonstration			
Events	Pre – Implementation/ Software demonstration	**Actors**	User2
Diary Observation (translation, notes)	• Creating a live scenario for the software demonstration • The client had some issues with their existing software, as noted in the sales team report, such as their inability to follow the order, missing orders, and difficulties with the inventory of items. • Analysts explained what the software could do in order to solve the client's issues. • Analysts linked the product functions to business process by explaining some of the business process cases to show how product functions cover the client's business.		
Meeting Notes	**Pre - Implementation**	**Organization 1**	**Inventory and purchases**
	• Analyst Clarify client issues at the start of the meeting. • Analyst uses the sales team report to ask about the client issues. • Analyst explains that the software will help to solve his issues. • Analyst presents the software main parts. • Analyst start by the setting functions for the software to be running. • Analyst explains the software functions such as add new item, items level of categorization. • Analyst discusses the inventory business process • Analysts uses a real case to explain the software functions such as add item and make item order through the software.		
\Descriptive analysis	I attended a software demonstration meeting during which the analysts aimed to show their product to potential clients and to convince them that the software product they had to offer would satisfy the client's needs. "We represent our software to a client by developing a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment. It really helps us to explain our software functions and connect these functions to a real case [team leader]". The clients run a goods inventory and they currently have some software that they use in the company. However, their current software is not adequate for their needs and it currently has some issues. This causes problems like the staff of the inventory having problems following an order, missing orders, and difficulties relating to the inventory of items. All of these issues had been made known to the analysts through the sales team report. The analysts went into the meeting with the purpose of telling the clients about their own inventory software which they think will solve the client's issues. The analysts did a few different things during this meeting. At the start of the meeting, first they clarified the client's issues. They kept referring back to the sales team report in order to ask about the client's issues. They then began explaining what their software could do to solve those issues faced by the client. They gave an initial description of their software product's functionality. It seemed that they did this in order to give the clients some kind of vision about the product's functions, and in order to increase the client's participation in the discussion. But the main thing that the analysts did to try to convince the client that their product was the right answer for them was to incorporate a live scenario to use during the software presentation. So they didn't just describe the product and its functions but actually demonstrated it live for the client, giving it a kind of test case and showing that it could do the work the client needed. This meant the analysts actually went into the client's inventory and began processing items orders using the software they had made. Actually, this live scenario seemed to have two different purposes: one was to try to convince the client to buy the product, but the other was also to use the live scenario to help analysts collect more information about client needs. For example, while using the software in a live scenario, they could show the clients the software while it was actually running, but they could also gain a better immediate understanding of the problems the client was having – and then become better informed about how to identify possible solutions. The analysts first set up the software functions that needed to be running, then described the various software functions, and then demonstrated the software functions as they worked. This included showing the client features such as 'add new item' and showing them levels of categorization available. They then used a real case scenario of trying to place a items order so they could show how the software would let the client 'add item' and make an item order. For example, they showed the client the interface for making items orders, and they showed the client what would happen. The client actually wanted to have a choice where particular items orders were sent to and therefore needed to have reports in the system. During the live scenario the analysts showed that the software had this capability. The whole time that the analysts demonstrated the software they		

	kept on linking the product's functions to the client's business process by connecting some of the client's concerns about business process (i.e. things that the client said had gone wrong before and various things that they wanted the software to do) to what the software could do. They also discussed the client's inventory business process with the client. The end result was that they managed to show the client that the product functions could cover the client's business. There were people present on the analysts' team and. The people sent by the client were the manager of the inventory. During the whole time, the client seemed pretty happy or impressed by the software and at the end of the demonstration the analysts suggested that they thought the client would probably buy it. The tone of the meeting was very business-like and the analysts made sure to be really clear and efficient.

4.2.3. Immediate Interpretations of Events

Walcott (1994) also discusses a number of ways to approach "analysis". Walcott suggests that "an inherent conservatism and caution" is associated with analysis and that analysis generally shows some restraint (p. 23). While Walcott explains that the word "analysis" is usually accorded a broad meaning – simply the process of "transforming data" – he uses the term "analysis" more specifically to refer to "systematic procedures followed in order to identify essential features and relationships consonant with the descriptors noted above" (p. 24). He suggests that "analysis" is the more factual, hard scientific part of qualitative research, and that no matter what kind of analysis is conducted, if rigorous rules and procedures are followed during analysis, an analysis can be trusted; "analysis is the more orderly, less speculative side of data transformation" (p. 27).

Walcott suggests a number of different ways to approach analysis. One analysis strategy is to (1) "highlight your findings" (p. 29). During analysis, a researcher's decisions regarding what to focus on become increasingly selective. Smaller amounts of data will draw most of the attention. Here, the researcher could highlight material previously presented, or present the material again with finer levels of detail. The analysis will rely on "reports or summaries": only "facts" are presented, not a story (p. 30). The second strategy is to (2) "display your findings". In this approach, graphic representation of findings (charts, diagrams, and figures) may be used rather than prose. Photographs can also be used as "visual facts" (p. 31). The third strategy mentioned by Walcott is to (3) "follow and report 'systematic' fieldwork procedures", such as those used by "ethnoscientists and ethnosemanticists" (p. 32). He recommends Spradley & McCurdy's The Cultural Experience (1972; 1988) and Werner & Schoepfle's Systematic Fieldwork

(1987a; 1987b). A fourth strategy is to (4) “flesh out whatever analytical framework guided the data collection”. This form of “narrative technique” includes “componential analysis, consensus analysis, content analysis, discourse analysis, analysis of social settings” (p. 33). Another strategy is to (5) “identify patterned regularities in the data”. Here, the researcher discusses relationships between different elements of data and “what-goes-with-what” (p. 33). The sixth strategy Walcott mentions is to (6) “compare with another case”. As long as the comparative basis is appropriate, “controlled comparison between a known case and the unknown case being analyzed offers a way for the analyst to exercise control” (p. 33). The seventh strategy is to (7) “evaluate” (i.e. compare with a recognized standard)”. Walcott explains that “evaluation is a form of comparison in which some explicit or implicit standard supplies the comparability by which judgments can be made” (p. 33). An alternative along the same lines is that the researcher presents “how those immediately involved or affected evaluate what is going on” (p. 34). Walcott suggests that the qualitative researcher can also (8) “contextualize in a broader analytical framework” “through informed references to some recognized body of theory in one’s special field” (p. 34). A ninth approach is (9) to “critique the research processes”. In the case that the analysis cannot provide any strong level of certainty, the researcher can admit that they have had to engage in some speculation, yet stress that their observations do seem to have specific implications. An alternative is to “focus your analytical attention on the research process itself, becoming your own critic but drawing attention to your methods rather than to your results”, and suggesting what was learned from the experience (p. 34).

The approaches I have used to present and discuss my analysis of the data are also broadly similar to some of those advocated by Walcott (1994). For example, I have chosen, during my analysis, to highlight some of the findings mentioned in my day-to-day descriptions [Walcott’s number ‘1’], and sometimes move on to discussing those findings in more detail. I have also displayed many of my findings through graphic representation such as figures, lifecycles, charts, and appendices, and have supplied some photographs of the software organisations’ work environments [Walcott’s number ‘2’]. Perhaps most importantly, I have sought to identify regular patterns in the data that might begin to explain why analysts favoured particular actions or carried out practices in a particular way, and I have established and discussed some of the linkages between different elements of the data I collected [Walcott's number '5']. I have also made sure to contextualize my research and findings by making frequent references to previous research into packaged software, requirements engineering, and SMEs, so that my research is presented in proper relation to relevant literature and my findings discussed in relation to recognized bodies of theory [Walcott's number '8'].

Weber (2010, p.3) defined theory as "a particular kind of model that is intended to account for some subset of phenomena in the world. A theory is a social construction. It is an artefact built by humans to achieve some purpose. It is a conceptual thing rather than a concrete thing". Gregor (2006, 2007) examines five ways in which the term "theory" has been used in the literature: (1) theory for analysing, (2) theory for explaining or understanding, (3) theory for predicting, (4) theory for explaining/ understanding and predicting (EP theory), and (5) theory for design and action. This study fits the 'explaining and understanding' theory class. According to Gregor (2006, p.9), "theory for explaining" is suitable when the researcher uses an interpretive paradigm that refers to where – when – how - why events occur. However, this theory is formulated in such a way that making testable predictions about the future is not of primary concern.

Theory for understanding is used for conjectures that are "drawn from a study of how and why things happened in some particular real world situation, these conjectures could form the basis of subsequent theory development, or be used to inform practice" (Gregor, 2007). Research approaches that can be used to develop this type of theory include case studies (Yin, 1994), surveys, ethnographic, phenomenological and hermeneutic approach (Denzin & Lincoln, 2005), and interpretive field studies (Klein & Myers, 2001).

As previously mentioned, the research phenomena of interest in this study are understanding PSIRE at SPSVs and answering the question: "What are the analysts' practices in the context of packaged software implementation by small packaged software vendors in Jordan?" Another reason for why this research is directed towards an explaining and understanding theory is the previously identified need to understand the "why and how" behind such practices. Table 4.2.3-1 below provides an example of how the immediate interpretations of events and discussions of analysts' practices contained in this thesis contribute toward building a theory for explaining and understanding PSIRE (for further examples, see Appendix D).

Table 4.2.3-1 immediate interpretations of events

Situation (where)	Process (when)	Practice (How)	Causal Explanations (why)
Pre-Implementation	Software Demonstration	Explaining product functionality Initial explanation	• To give an initial explanation about the product. • To help clients create vision about the product functions, and to increase clients' involvement and participation in discussion • To present the functionality of the software.
		Uses the sales team report	• To clarify client issues • Explained what the software could do in order to solve the client's issues.
		live scenario	• To help enhance understanding of the problem, and simultaneously identify possible solutions. • To convince the client that the software product they had to offer would satisfy the client's needs • To develop a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment
		Link the product functions to business process	• To present the functionality of the software and identify its need according to the business process.
		Discuss the business process	• To understand clients' needs and their business process

4.2.4. Higher-level Theoretical Concept of Analysts' Practices

The third major aspect of qualitative description Walcott discusses is "Interpretation". When conducting "interpretation" "the researcher transcends factual data and cautious analyses and begins to probe into what is to be made of them". Interpretation is therefore distinct from "analysis" (p. 36). One problem Walcott identifies is how far beyond a case itself interpretation should try to reach. "Interpretation is not bound to the descriptive account as tightly as analysis […] but that does not free the researcher to float away with no discernible link to the case at all" (p. 37).

Walcott suggests a number of ways to approach interpretation. A researcher could (1) "extend the analysis". If a researcher wishes to be cautious, they could state the implications that could be seen in their data or drawn from their analysis, without explicitly drawing that particular conclusion or implication themselves (p. 40). A second possibility is to (2) "mark and then make the leap", noting the kind of uncertainty mentioned above but embracing the uncertainty, and speaking of findings or possibilities by way of words such as "inference", and following "inductive reasoning" "where the conclusion goes beyond the content of the premises and therefore cannot be stated with certainty" (Carnap, 1953, quoted in Sarana, 1975, p. 3) (p. 40). Walcott also suggests that if a researcher has a problem with successfully interpreting their data, they should "take the account as far" as they can with confidence, then stop: "a weak interpretation is better than no interpretation at all" (p. 41). A different approach is (6) to "turn to theory" (p. 43). Theory may not only assist analysis, but also aid interpretation, especially for linking case studies with larger issues. A seventh option is (7) that rather than using theory as an interpretive framework, the approach consists of actually developing such a framework, and possibly testing it on original data. Another option is to (8) "connect with personal experience". An interpretation can be personalized (stating "this is how I see it all") or made personal (such as by saying "this is how the research experience affected me") (p. 44). This second approach was referred to by John Van Maanen as the "Impressionist Tale" (Van Maanen, 1988, pp. 101-124). Walcott suggests that the best way to balance the three ways of presenting qualitative work – description, analysis, and interpretation – always depends on "the purposes of the research" (p. 46). According to Walcott, "description, analysis, and interpretation are the three primary ingredients of qualitative research. And they too are variously combined to achieve particular purposes. No single combination can be regarded as best, nor is a researcher required to include all three" (p. 49).

My interpretive methods align with those Walcott suggests a qualitative researcher could use. In the interpretations I offer here I have sought to build upon points I made in my initial analysis (my immediate interpretations of events), after engaging in further reflection, validation of results, and reference to past literature. While I have mostly constructed theory and a theoretical framework based on definite observations and recurring themes and events, I have also provided some interpretations that go beyond what can be stated with absolute certainty. However, I have always signposted those points where I engage in speculation. Lastly, as Walcott also advises, I have not sought to rigidly allocate equal amounts of attention to the different dimensions of presenting qualitative research – description, analysis, and interpretation – but have combined the three approaches in the way that I perceived best suits the purpose of the research, best suits the ethnographic approach used, presents my findings in the most effective and understandable way, and best allows me to answer my initial research questions.

Interpretations were conducted to derive higher-level concepts that would comprise the theoretical constructs in PSIRE to reach a higher level of abstraction (Khoo & Robey, 2007). For example, “live scenario software demonstration” was conceptualised as a strategy that was intended to help analysts convince clients about a software solution, as they demonstrate and discuss a possible solution. Each process or practice is a general name for specific instances of the phenomenon as being part of a conceptualised grouping and is described via an analysis of these respective concepts and by the study of some theories taken from previous literature in this field.

Figure 4.2.4-1 below provides graphic representation of various practices used by analysts during PSIRE and shows when they occur during PSIRE: either in the Pre-Implementation phase, or used During Implementation. The figure also demonstrates that there are different major processes involved in each phase, such as a ‘Feasibility Study’ and a process of ‘Identifying Misalignments’, and processes of ‘Installation’, ‘Assessments’, and ‘Software Demonstration’. The practices used during PSIRE are also aligned with the process that they assist and/or inform.

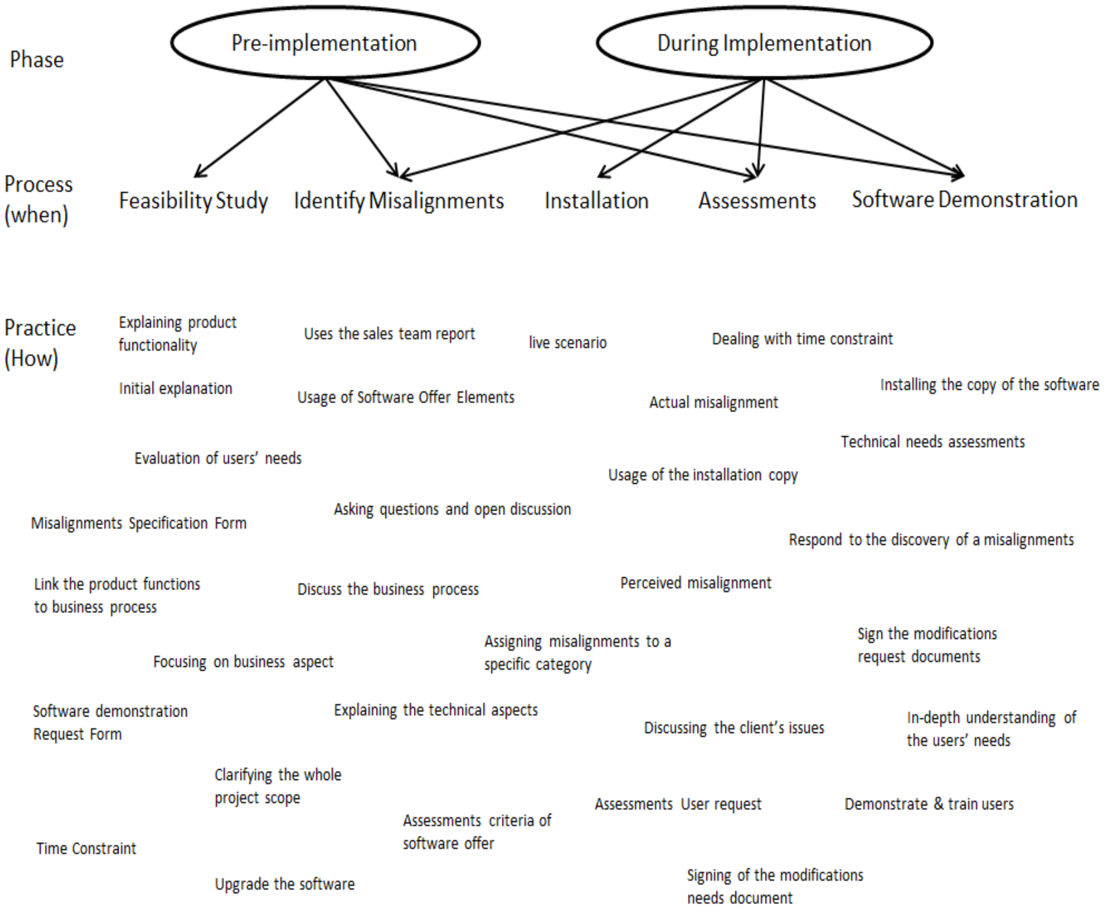
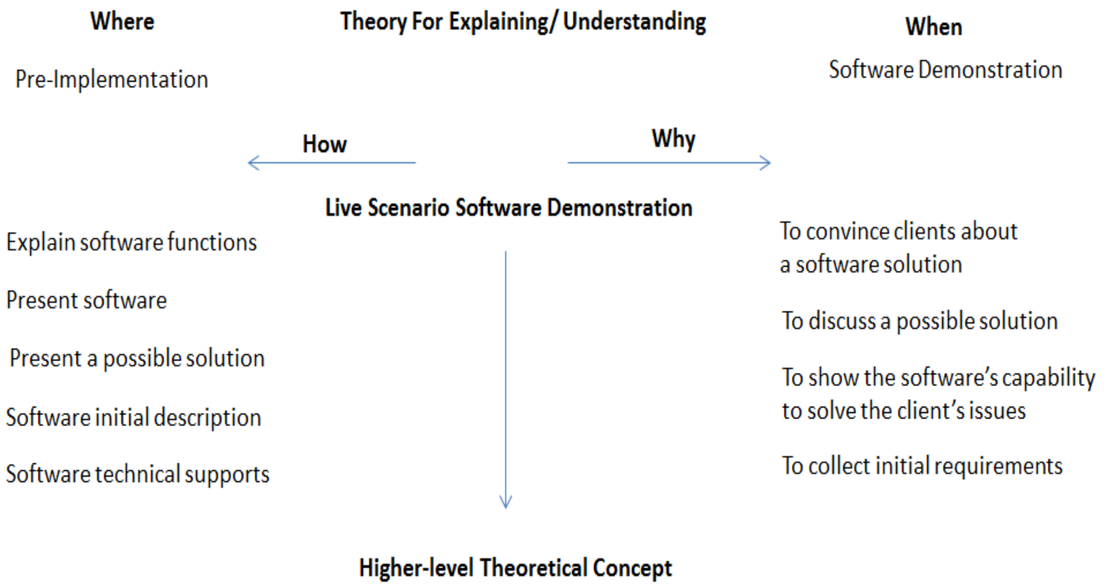

Phase
Pre-implementation
During Implementation
Process (when)
Feasibility Study
Identify Misalignments
Installation
Assessments
Software Demonstration
Practice (How)
Explaining product functionality
Uses the sales team report
live scenario
Dealing with time constraint
Initial explanation
Usage of Software Offer Elements
Actual misalignment
Installing the copy of the software
Technical needs assessments
Evaluation of users' needs
Usage of the installation copy
Misalignments Specification Form
Asking questions and open discussion
Respond to the discovery of a misalignments
Link the product functions to business process
Discuss the business process
Perceived misalignment
Focusing on business aspect
Assigning misalignments to a specific category
Sign the modifications request documents
Software demonstration Request Form
Explaining the technical aspects
Discussing the client's issues
In-depth understanding of the users' needs
Clarifying the whole project scope
Assessments User request
Demonstrate & train users
Time Constraint
Assessments criteria of software offer
Upgrade the software
Signing of the modifications needs document

Figure 4.2.4-1 Graphic representation of various practices

Figure 4.2.4-2 shows how discussions of analysts' practices contained in this thesis contribute toward building a theory for explaining and understanding PSIRE (for more examples relating to 'live scenario' software demonstrations, see Appendix G).


Where
Pre-Implementation
Theory For Explaining/ Understanding
When
Software Demonstration
How
Why
Live Scenario Software Demonstration
Explain software functions
Present software
Present a possible solution
Software initial description
Software technical supports
To convince clients about a software solution
To discuss a possible solution
To show the software's capability to solve the client's issues
To collect initial requirements
Higher-level Theoretical Concept

'Live scenario software demonstration' was conceptualized as a strategy that was intended to help analysts convince clients about a software solution, as they demonstrate, and discuss, a possible solution.

Figure 4.2.4-2 Graphic representation of various practices

4.2.5. Limitations to the Investigation of Events

During my research the information gathered and analysed pointed out that some events in the organisations need to be investigated in more detail. For various reasons I did not have the possibility to engage in further in-depth investigation of some of these events. For example, I learnt that analysts were writing a report to evaluate the time and cost efforts that would be involved in customising a software package. However, the participants chose not to share with me any specific information related to writing the report and its influence on the PSI. Since I was not allowed to participate in this part of the process, I could not make a sound evaluation about the relationship between writing the report, its content, and PSI. However, it is likely that the steps involved in actually writing the report or the template of the report itself is not as significant to the outcome or process of PSI as the practical work that underpins the actual production of such a report.

One more part of the process of gathering the users' initial needs that I was not able to investigate in any detail was the use made of sales team reports. From what I could gather, sales team reports provided the analysts with information about the user's organisational structure and size. However, the analysts themselves did not believe that PSIRE starts before the demonstration takes place and asked me not to bother the sales team with PSIRE questions. Since permission to interview the sales team or seek more details about the sales team report was not granted, I had no choice but to omit the investigation of this theme. Table 4.2.5-1 presents information relating to the reasons why some themes were not investigated further in the current research.

Table 4.2.5-1 Limitations to the investigation of codes and themes

Where	**When**	**How**	**Why**
Demonstration presentation	Pre-Implementation	Evaluation report	➢ To evaluate the effort of cost and time required customising the software
Limitation on research	➢ The participants did not agree to discuss the information written in their analysts' report. I received a high level description of it, but this was not enough to let me understand the report content and how it related to PSI.		
Demonstration presentation	Pre-Implementation	Sales team report	➢ To provide users' organisation structure and size
Limitation on research	➢ Participants asked me to avoid interacting with the sales team in terms of questioning them about elements of the requirements determination process that happened before demonstration presentations occurred. They said that managing software specificity and generality in packaged software started from their demonstration presentation, and that what they got from the sales team was not important. While I did not agree with this, I could not conduct further investigations without permission. However, permission to interact with the sales team was never granted.		

4.3. Theory and Generalisability

The conceptual definitions of theory and generalisability are controversial topics in the information systems literature. Researchers have offered a range of viewpoints on the validity of generalisations in qualitative research or methods used to enhance generalisability (Klein & Myers, 1999; Gregor, 2006, 2007). The definition of 'theory', what it involves and what it can do have similarly been explained in a range of ways (Klein & Myers, 1999; Gregor, 2006, 2007).

Lee & Baskerville (2003) clarify the concept of generalisability in IS and present “a framework for classifying its different forms”. Lee & Baskerville state that generalisability refers to “the validity of a theory on a setting different from the one where it was empirically tested and confirmed”. “The generalisability of an IS theory to different settings is important” not only for research but also “for purposes of managing and solving problems” faced by corporations and organisations.

Lee & Baskerville (2003) go on to stress that generalisability does not have to rely on quantitative data, on sample bases, or on statistics – researchers engaged in qualitative research can make claims about generalisability that are suitable to their own research. Induction can involve “reasoning from data points in a sample to an estimate of a population”, even though, as identified by David Hume, there are problems involved with such induction.

Following from Hume’s infinite regress theory, acceptance of which would seem to mean that “we cannot generalise at all (Campbell & Stanley 1963)” (Lee & Baskerville, 2003), Lee & Baskerville note that increasing the size of a sample or the number of test cases may improve the reliability of a study (meaning that the procedure used could be reapplied by the same researcher or another researcher), but it cannot increase “the generalisability of a study to its population” (Lee & Baskerville, 2003). This is because we cannot make claims that transcend the available data. Lee & Baskerville (2003) identify this as one of the limitations of statistically-based induction.

According to Lee & Baskerville (2003) generalising can occur in four different ways: “from empirical statements to other empirical statements, from empirical statements to theoretical statements, from theoretical statements to empirical statements, and from theoretical statements to other theoretical statements”. Yin (1984, 1994) had earlier made similar observations, explaining that researchers can “generalise from a sample to a population, from experimental subjects to experimental findings, and from a case study’s findings to a theory” (Lee & Baskerville, 2003).

Lee & Baskerville (2003) build their framework on the four kinds of generalising identified above, referring to them as Type EE (Generalising from Data to Description), Type ET (Generalising from Description to Theory), Type TE (Generalising from Theory to Description), and Type TT (Generalising from Concepts to Theory).The first type, EE, is sometimes used in ethnography, but when data is collected in this way, especially via interviews and speaking, the ethnographer has to continually assess the believability of the information harvested. If their assessment suggests to them that the information is believable, then the data receiving a favourable assessment may be “generalisable to a valid descriptive statement”. However, the researcher cannot suggest that the descriptive statement thus constructed is generalisable “beyond the domain that the researcher actually observed”. This would remain true even if the researcher increased the number of people interviewed. All such an increase would achieve is establishing that the data obtained can be generalised to the findings of that particular experiment. The researcher still cannot generalise about parts of the domain that they have not observed (Lee & Baskerville, 2003).

Most pertinent to my thesis is generalisability Type ET, ‘Generalising from Description to Theory’. Here, the researcher generalises from empirical statements to theoretical statements (Lee & Baskerville, 2003). This type can involve two kinds of generalising: “the generalisability of measurements, observations, or other descriptions to theory, and the generalisability of the resulting theory beyond the sample or domain that the researcher observes” (for example, parts of an organisation where he/she has not conducted interviews or collected data) (Lee & Baskerville, 2003).

Yin (1984, 1994) also writes about this form of generalisation, including generalising from “experimental findings to theory, generalising from case studies to theory, and generalising from population characteristics to theory” (Lee & Baskerville, 2003). Yin focuses especially on case studies and suggests that the empirical descriptions in a case study can be generalisable to a valid theory if the researcher has been careful to include multiple sources of evidence, use a case study database, and engage in member checking. Such care would ensure quality descriptions. However, the difficulty still remains of whether it is possible “to claim that a theory will remain valid beyond the observed case”, because this seems to require accepting “the uniformity of nature proposition” that Hume finds issue with (Lee & Baskerville, 2003).

The interpretive IS researcher Walsham has expressed approval of Yin’s work, yet rather than suggest (like Yin) that the empirical descriptions in a case study have no generalisability beyond their given case, “Walsham explains that beginning with facts or the rich description of a case, the researcher can generalise to concepts, to a theory, to specific implications, or to rich insight” (Lee & Baskerville, 2003). Klein & Myers (1999) also accept the possibility of generalising from empirical statements to theoretical statements and refer to it as “the principle of abstraction and generalisation” (Lee & Baskerville, 2003).

Lee & Baskerville (2003) also mention that Glaser & Strauss’s (1967) grounded theory also suggests that “theory is grounded in descriptive categories and relationships that emerge from properly collected and coded data”. They note that this is the same phenomenon as generalising from empirical statements to theory, and that Eisenhardt (1989) accepts the validity of both Yin’s (1984, 1994) case study method and Glaser & Strauss’s (1967) grounded theory.

Lee & Baskerville (2003) observe, therefore, that “the notion of the generalisability of empirical observations to theory is well developed” and that “criticisms that case studies and qualitative studies are not generalisable would be incorrectly ruling out the generalisability of empirical descriptions to theory” (Lee & Baskerville, 2003).

The final conclusion by Lee & Baskerville (2003) is that “in a case study, the researcher may appropriately strive to develop a theory that is generalisable within the case setting” and that it would not be appropriate to criticise such “a theory for a lack of generalisability to other settings”, because whether dealing with qualitative or quantitative research, “there is only one scientifically acceptable way to establish a theory’s generalisability to a new setting: It is for the theory to survive an empirical test in that setting” (Lee & Baskerville, 2003). They argue that “if there is a quality of case studies that might merit criticism, it would be a lack of 'particularisability'" – i.e. that the theory cannot be transferred to descriptions of another setting (Lee & Baskerville, 2003).

4.4. Evaluation of the Research

An explaining and understanding theory (Gregor, 2006) aims at explicating how, when, where, and why events occurred. All alternative explanations should be scrupulously examined and assessed for internal validity. Justification for the contribution of knowledge provided by this type of theory is made primarily on the basis of whether new or interesting insights are provided, as well as whether or not the outcome of the study seems plausible, credible, and consistent.

Schultze (2000) explains that the main criteria for evaluating the quality of ethnographic work in general were established by Golden-Biddle & Locke (1993). According to Golden-Biddle & Locke, there are "three dimensions central to writing a convincing and publishable ethnography. These were authenticity, plausibility, and criticality" (Schultze, 2000). What Golden-Biddle & Locke refer to as 'authenticity' has, however, often been expressed by other theorists in terms of 'reliability' and 'validity'. Although an ethnographic study cannot be replicated, Schultze (2000) suggests that in ethnography, reliability may be established if readers feel that the ethnographer has been reasonable in how they interpret events. Therefore, this study utilises a consistent process for data analysis which is fundamental to determining the credibility of findings. At its core, the approach supports the transformation of raw data material into themes and categories. There are several possible ways that this process could be applied, but the fundamental rule in using this approach is to describe the process in all its possible details, so that the reader will agree that the final outcome generated and the results extrapolated from the data collected seem plausible (Gregor, 2006).

Moreover, the philosophical foundation of the interpretive approach in this study was supported by multiple constructions and interpretations of the events. Using multiple methods of data collection and having various sources for the collected data provided the data with depth and richness and reduced bias. Credibility and authenticity were achieved by presenting experiences and descriptions of the events by both the participants and the researcher. In cases where it was suitable, a quotation from the participants was presented. Otherwise, the events were described with subtle details adopted from participants' perspectives and activities.

'Validity', meanwhile, relates to "the representativeness of the data and the truthfulness of an ethnographer's interpretation" (Schultze, 2000). Validity in ethnography can be assisted by the ethnographer's collecting large amounts of different kinds of data. To achieve validity within this current research, data collection took place over a period of 7-8 months. This was a long enough period to build a rapport with participants and gain their trust. This, in turn, increased the rigor and trustworthiness of the findings. The data collection methods were: interviewing the participants, participants' observations, and focus groups. However, the data collected by these methods has been expressed through descriptive analysis notes which reflect a day-by-day analysis of analysts' practices (as shown in Appendix D). Collecting data via multiple methods allowed more reflexivity in the analysis of the data and assisted understanding of the data.

'Plausibility' and 'criticality' are other factors that need to be considered when evaluating the quality of an ethnographic study. Plausibility in ethnographic analysis can be assisted by adhering to elements of the academic article genre, by justifying the research presented and how it fills gaps or creates a new understanding, and by aligning findings with common experiences (Schultze, 2000). The research must be made "relevant to the concerns of the intended audience". Schultze notes that in her own study, she "enhanced plausibility by motivating the need for practice-oriented research", "by pointing to calls for such research in the literature". Criticality can be obtained by "challenging readers to pause and think about a specific situation", provoking readers to answer questions, or "guiding readers through imagining ways of thinking and acting differently" (Schultze, 2000).

Schultze's main conclusions are that knowledge work has "three informing practices: expressing, monitoring, and translating" (Schultze, 2000) and that what underlies each of these practices is the knowledge worker's aim "to balance subjectivity and objectivity" (Schultze, 2000). This study has aimed to follow methods such as those outlined by Schultze (2000) and (Gregor, 2006, 2007) in order to maintain the validity and rigour of the study. In Table 4.4-1a-c below, I have adopted Schultze's (2000) criteria for assessment of the quality of my own 'ethnographic' contributions to this research.

Table 4.4-1a **Comparison of this research against the evaluation criteria for ethnographic research**

Evaluation criteria for ethnographic research	
Authenticity	
Everyday life as lived by members of the field;	Chapter 5 provides realistic descriptions of aspects of the analysts' practice lives, my own daily descriptive analysis augment this material
Vernacular of the field;	Practices, requirements engineering, and technical vernacular are used
What members think about their lives in the field;	Chapter 5 & 6 provide frequent reflections upon the analysts' practice lives, my own daily descriptive analysis forms & Immediate interpretations of events analysis augment this material
Who the ethnographer talked to and observed;	Actors and their roles are made explicit in the analysis of Chapter 3 section 3.4 and my own daily descriptive analysis forms
The nature of the researcher's relationship with various categories of people in the field;	Explicitly defined in Chapter 3 section 3.3.6 & 3.3.7
The response of others on the scene to the researcher's presence;	As a participant observer, I was not seen as an 'outsider' as a classic ethnographer might. However, Chapter 3 section 3.3.6 & 3.3.7 and Appendix D provide the challenges I faced in term of culturally sensitive issues and data I could not discuss
Researcher's pre-understandings of the studied scene;	The prior work experience of the researcher as an analyst could be considered as providing a base of pre-understanding.
Researcher's interest in the scene;	Motivation for the study is explicitly outlined in Chapter 1

Table 4.4-2b **Comparison of this research against the evaluation criteria for ethnographic research**

Authenticity	
Researcher's length of stay; start and end dates of the research;	The method section describes that I was in the field for 7 months. However, the rich data has been collected from Feb to April 2012
The relationship between the field notes and the written up ethnography;	In discussing my own informing practice of translating, I describe how I used my field notes during my data analysis phase, and I used description forms for the field notes. The field notes constitute raw data to the study, and are featured in most chapters, either as support for argument based on reflections or as data for analysis as in Chapters 5, 6 & 7.
Presenting "raw data" like field notes, documents, and transcribed interviews;	Extensive excerpts from the field notes are presented throughout the study, esp. in Chapter 5, Chapter 6, and Chapter 7.
Plausibility	
Adhering to academic article genre;	This manuscript adheres to the doctoral thesis genre in that it follows a fairly standard thesis structure, formatting conventions, and few conferences and journal publications
Justifying the research and differentiating its contribution through the identification of gaps in our understanding or the development of a novel theoretical approach;	Chapter 1 has presented the justification for the work, and summarised the contributions arising from the study. Chapters 5 and 7 have presented the novel theoretical contributions of PSIRE through a focus on analysts' practices.
Normalizing atypical research conditions and aligning the findings with common, everyday experiences.	The descriptive analysis for each process in Chapter 5 and 6, serves to normalise the research in the everyday context of the actors.
Criticality	
Cultural juxtaposition;	By positioning myself as an analyst and comparing and contrasting my practices with the participants; and institutional cultures.

Table 4.4-3c **Comparison of this research against the evaluation criteria for ethnographic research**

Self-revealing account	
Use of personal pronouns;	“The use of "I" is pervasive in the descriptions of my own informing practices as well as in the excerpts from the fieldnotes; this consistently highlights my role as narrator” (Schultze, 2000)
Age, gender, and race, epistemological assumptions and theoretical point of view;	To give readers a sense of who they are following through the field, I present myself as a single, 30 year old man; I do not mention my race and cultural background, as these seem irrelevant to the subject matter and my relations with participants in the field as I was of the same race as the participants;
Disclosing details that present an unflattering picture of researcher, e.g., mistakes made;	In both the method section and the description of my informing practices, I discuss problems with employing such a high-risk strategy of data collection; Chapter section 3.3.6, 3.3.7 and 3.10. I give examples of mistakes I made with respect to the type of the data that I could collect and discuss, and the data collection approach in regard to interacting with participants;
Rendering canonical the problematic and less-than optimal research conditions.	While I faced a challenge in terms of restrictions being placed on the types of questions I could ask within the organizations and the types of data I could discuss (see section 3.3.7), I found that following the kinds of analytical approaches for qualitative research advocated by Walcott (1994) and Sandelowski (2000) was beneficial
Interlacing "actual" and confessional content	
Interlacing self-reflexive and autobiographical material with "actual" ethnographic material;	“I avoided the trap of constructing a purely methodological and self-absorbed account of my trials and tribulations as an ethnographer;” (Schultze, 2000)
Limiting autobiographical material to information that has relevance to the subject of the research.	I “did not elaborate on my race and cultural background as these did not appear relevant” (Schultze, 2000). However, broader ‘cultural’ dimensions have been impacted by cultural sensitivity issues related to the Jordanian context in the study

4.5. Summary

The data collected during my ethnographic study was primarily qualitative and was analysed using a congruent data analysis strategy. By following an inductive approach, I allowed theory to emerge from the raw data. Data analysis must be carried out in a way that ensures that the analysis will be valid and that findings will be credible. The data collected in this study initially via participant observation was put through three stages of inductive analysis, during which I found that first low-level codes. The initial raw data was coded inductively, using key-point coding. Initial codes were developed, based on a frequency-based approach. While I faced a challenge in terms of restrictions being placed on the types of questions I could ask within the organisations and the types of data I could discuss, the approach I followed was similar in kind to the types and arrangement of approaches advocated by Wolcott (1994). Wolcott, writing about transforming qualitative data into useful insights, posits that qualitative research involves three major and distinct subsections or phases, within which researchers may take up different approaches to presenting data. He refers to these wider phases as "description", "analysis", and "interpretation". The methodology I followed also divided my qualitative analysis into these three different forms, as indicated in Figure 4.2-1 above. Through field notes detailing events and participant observation and day-by-day descriptive analysis, I fulfilled the "description" form of qualitative inquiry discussed by Wolcott. My immediate interpretations of events and day-by-day descriptive analysis presented facts relating to what I had observed and then sought to identify interrelationships between different events, ideas, and processes involved in PSIRE.

In terms of practices of generalisation, my study could be said to follow the practice of 'Generalising from Description to Theory' (generalisability Type ET) discussed by Lee & Baskerville (2003). Here, the researcher generalises from empirical statements to theoretical statements (Lee & Baskerville, 2003). In this thesis I have aimed to put forward new theory for 'explaining', such as discussed by Gregor (2006), in order to explain 'when – how – why' events occur, and theory for 'understanding' (Gregor, 2007) that builds on conjectures about what events happened and can be used to inform practice. The discussions of RE practices contained in this thesis contribute toward building a theory for explaining and understanding PSIRE and are accompanied by my provision of a business process flow chart identifying where PSIRE occurs in organisations (see Appendix C) and (in Chapter Seven) a Star Diagram showing the Parallel Processes involved in PSIRE.

Chapter 5 Findings & Results: Analysts' Practices

5.1. Introduction

The previous chapter presented the data analysis approach utilised in this study. This chapter explains how the chain of results was developed through application of the data analysis strategies described. In particular, this chapter provides a detailed account of the application of the chosen analysis strategies and how this allowed the theory to emerge.

This chapter draws on data collected from the software companies from December 2011 - June 2012, and is divided into two parts. In the first part, I describe with ethnographic detail the course of events taking place when Packaged Software Implementation Requirements Engineering (PSIRE) was enacted. In doing so, I address a pre-implementation stage in PSIRE that includes (1) a software demonstration request, (2) the mechanisms of a software demonstration, (3) the mechanisms of scoping, (4) the development of a software offer, and (5) assessment of pre-implementation. The second part of this chapter addresses a detailed analysis of analysts' practices and activities in PSIRE, based on an ethnographic account. In particular, I address elements of the implementation stage that include (1) the mechanisms used to identify misalignments, (2) the mechanisms of assessment of misalignments, and (3) responding to users' needs and to misalignments.

It should be noted that this study presents the first account of packaged software implementation with a focus on requirements engineering practice within SPSVs. This chapter provides not merely a chronological description of activities and events, but rather a theoretically informed explanation of how actions and events unfolded over time and led to a particular outcome.

Section I: Pre-Implementation

The pre-implementation practices in this study resemble such feasibility studies as those used in RE practices *at a high abstract level* and so involve dealing with software objectives, time, budget, project scope and domain solutions.

[They] identified scope creep as one of the major factors contributing to the failure of packaged software implementation [Al-Mashari & Al-Mudimigh, 2003]
"Organizations should reduce the implementation time frame because it decreases the opportunity for implementing customisations" [Haines, 2009]

In this section, I address a pre-implementation stage in PSIRE that includes (1) a software demonstration request, (2) the mechanisms of a software demonstration, (3) the mechanisms of scoping, (4) the development of a software offer, and (5) assessment of pre-implementation.

5.2. Software Demonstration Request

In packaged software implementation, sales and marketing teams and analyst teams work together in order to organise and provide software demonstrations to potential clients. In this section, I discuss the mechanisms of how these teams interact to set up and run software demonstrations.

5.2.1. Search for Potential Clients

The organisations participating in this study both have an internal marketing department looking for potential clients on a daily basis. They may also find clients via personal networking carried out by individual staff members of any department within the company. For example, the internal marketing department of Organisation One has created an internal database resource that provides information about clients to other departments within their company, such as sales, analysts, top management and development. They are therefore able to find a variety of information about potential clients. The database contains records relating to about 25,000 potential clients. Of these 25,000 records, 12,000 are active, which means that the marketing team contacted those clients within the last three months. The potential clients' record status changes to inactive if the marketing team's last contact with them was over three months ago. The company developed their approach toward managing clients' information from a marketing perspective and they developed their own tool to manage such information.

It is a long process. Listen, first, we have our marketing team who call a lot of companies to find out if they are interested in any type of software that we provide..... We keep updating our database about potential clients. We got this information from newspapers, searches on the internet, different ministry websites such as 'education', 'manufactory' and so on… The important thing for us is to find out if they are looking for a software product to use. This of course depends on their kind of business… so we try to advertise our software product to them… most of the time they ask to send information about our software product so we tell the sales team to visit them…. Before such a visit happens we try to get as much information as possible about the potential clients, such as their business type [Marketing Manager]

In fact, I discovered that the marketing team usually begin the interaction with potential clients through phone calls and emails in order to arrange a meeting between them and the software sales team. The sales team then meets with the potential client in order to explain further detail about the software that their company provides. At these meetings, the sales team collects information about the potential client's company structure. After this, the sales team arranges a demonstration presentation about the software, which is to be carried out by the analysts. The sales team then completes meeting minutes and a customer analysis form that is used by analysts to prepare for their demonstration of the software. However, once more, the description of this process still leaves some questions about the process unanswered. For example, how do the customer analysis form, the client's company structure information, and the software demonstration request impact on how the analysts' team approach a demonstration presentation?

5.2.2. Software Demonstration Request form

Analysts in the companies considered in this study frequently receive requests for software demonstrations. The discussion below shows one such example.

Client Name: A
Address: AAA
Demonstration Request: Yes
Time and Date:
Request From: G
Client importance: high
Place:
Client Description: Client A. is a primary and high school that includes 60 employees and 3000 students. A. has about 21 users of the packaged software "School Management System (SMS)". Client A. has 5 departments that use SMS: Students' information, School clinic, School Library, School enrolment, and Financial department.
Client A. used a SMS at their school. However, Client A. is interested in replacing their SMS. The reason for this is that the current SMS does not support the central database. In other words, the five departments are not connected with each other through the SMS so they normally send their transactions relating to student profiles manually via paper documentation.
[Demonstration Request Form Information]

In this case, analysts received a software demonstration request via their sales and marketing teams. The request form that was used was filled in by the sales and marketing teams through the use of a software tool that the analysts' organisation had developed in order to facilitate the exchange of information and to manage work. The tool is used to frame the day-to-day work of the analysts' staff at the company site. Even without possessing any prior knowledge about the analyst organisation involved, one can note much about this interaction. For example, the request form is presented as a formal document that uses technical language and business language that is addressed to an audience familiar with this particular field of software functionality. For example, the sales and marketing team explain the client's business problem of a lack of integration and knowledge transfer between departments. Meanwhile, the form also mentions problems arising from the client's use of a decentralised database.

The form also mentions the client's issues, organisation structure, current status and the kind of changes to their system the client is seeking. However, there are some issues of importance that cannot be understood simply by reading such a text as the one above. For example, the text above leaves unknown what mechanism the analysts' team uses to respond to a software demonstration request. Additionally, it leaves unexplained how such a request form is created at the sales and marketing team site.

5.2.3. Response to Software Demonstration Request

The customer analysis form, client's company structure information and software demonstration request information help us to decide on software demonstration mechanisms... you know, it helps us decide what different issues impact on our choice and how the client's type impacts on our choice... for example: the number of users within a company lets us decide if we are going to demonstrate a VB.NET solution that is more expensive and serves a greater number of users, or a VB6 Solution that is less expensive and can serve only a limited number of users...sometimes clients have issues outside our work domain, so we just cancel the software demonstration...[Team Leader]

The response to a software demonstration request is prepared by the analysts. The analysts in one of the organisations I observed could choose to respond to such a request by selecting one of two different types of software solutions provided by a 'drop down menu': VB.NET solution and VB6 Solution. These solutions are linked to the services they provide. To help the analysts decide upon the solution and also the limitation of the work domain of the analysts' company, discussion meetings are held, involving the sales, marketing and analysts teams. During these detailed discussions, the teams carefully choose which solution to offer in the given circumstances.

The comment quoted above shows one analyst's perspective on the importance of various sources of information. With this example, I am not able to provide any detailed information about the meeting discussion since it is confidential. However, in general, the discussion during this meeting related to the client analysis form, the client's company structure information, and the software demonstration request information. The following points address how these factors impact upon analysts' decisions about which software product solutions to offer. I have represented the factors from an IS point of view.

- Clients' organisation size; the number of users, departments, transactions, and the relations between the transactions, number of database records, and level of security required.

- Clients' issues; the limitation of the work domain of the analysts' company, and the limitation of the software product solution. Sometimes analysts may not be able to solve some of the client's issues because they are beyond the scope of the product solutions.

Some factors relating to a business point of view and to sales perspectives, such as seasonal issues and financial problems faced by the software company, are beyond the scope of this study.

In this section I have discussed how analysts respond to the software demonstration request, and the factors that might impact upon their decisions regarding possible solutions. In the next section, I discuss the mechanisms of the software demonstration.

5.3. Mechanisms of Software Demonstration

There was strong consensus that software demonstration for packaged software should be considered as to convince the users that alternative solutions may exist, even if they do not serve as existing process surrogates or become (part of) the solution [General Manager]

5.3.1. Analysts' Roles during pre-implementation

The software we present is based on the notes from the sales team about the client's interest in potential software. Then we present the functions of the software that supports the client's business..... I think that helps us to convince the client to buy the software ....but how can we convince the client? We try to present the key functions that will benefit the client's business. You need to stress what is important for them, and show them that you understand their needs and that you are the right people to design software for them that satisfy their business process; so we explain the software functions and how these functions cover their business. In many cases the client may wish us to focus on their particular issues if there are any. So we try to show how the software could help them to solve their particular problem.... keep in mind, our software may have some features that are relevant to the client's needs and other features that are less relevant. Of course, once we demonstrate some features, the client may decide on some new requirements, thinking those requirements very attractive when they did not initially consider them to be requirements. We will perceive even these new requirements as part of the initial requirements [Team Leader Organisation one]

Various insights about software demonstration can be derived from the above comments if we take them as representative of what the whole team of analysts does during a software demonstration. For example, it can be discerned that analysts consider the importance of a software demonstration from two dimensions: one is the business dimension which consists of presenting a possible solution to the client's issues and convincing the client to buy the software; the other dimension relates to software analysis, as the client might recognise new requirements and features in addition to those they initially perceived as requirements.

Before, the sales team made a software demonstration. But we found out that they just talked without understanding the software so sometimes they mentioned the wrong things to our client. The sales team is useless when making a software demonstration because they did not develop the software [General Manager Organisation one]

One insight that can be derived from the comments above is that the analysts' roles have changed from only collecting requirements and software analysis to also engaging in the pre-sale of their software. In this company, the software demonstration was always carried out by the analysts. It seems that this was likely due to the understanding that analysts were the only ones who knew how the software is built, how it works, and how to explain it. It also appears that it was thought more likely that the client will buy the software if he/she feels the software solves their problems and helps them achieve their objectives – and the analysts may be better at explaining such issues than the sales team. Analysts approach the software demonstration from the perspective of convincing the client to buy the software, and from the perspective of identifying misalignments. This requires skills, knowledge and experience in packaged functions and how they relate to each other. It appears that in this organisation, at least, the sales team did not possess such knowledge and capabilities.

5.3.2. Software Demonstration utilising a Live Scenario

We represent our software to a client by developing a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment. It really helps us to explain our software functions and connect these functions to a real case [Team Leader]

In this section I draw on one observed case in which a team of analysts demonstrated the software "Restaurant management system" for clients. The marketing and sales team contacted a client organisation in order to set up a demonstration presentation that would be conducted by the analysts' team. In this case, the team of analysts started preparing for the demonstration presentation by using the sales and marketing team's report about the client's organisation structure and the initial issues involved. The analysts set up the demonstration presentation by creating a live scenario for the software demonstration. The reason behind this was that "Restaurant management system" software includes hardware functionalities, such as using a personal digital assistant (PDA) to take customer orders, and software functionalities, such as representing menu items, and the analysts believed that the system capabilities could be most effectively demonstrated with a live case.

The client had some issues with their existing software, as noted in the sales team report, such as their inability to follow the order, missing orders in the kitchen, and difficulties with the inventory of items. During the demonstration process analysts explained what the software could do in order to solve the client's issues. They listed a set of functions that the software provided, such as making it easy to take an order, making sure the kitchen receives the order, helping the cashier to receive the payment required, and creating a record of their being paid. The team of analysts then sought to represent these functionalities by using a live scenario that allowed them to link software functionalities to the business case. The team leader of the analysts explained to me that creating a live scenario of a business case through software demonstrations was usually capable of delivering the possible solution that the client needs.

During the particular case mentioned above, the client was facing an inability to follow food orders, and orders not being relayed to the kitchen. However, during the demonstration, the analysts used a PDA to place an order from a menu item on the client's actual menu, and then showed that the order had been sent to the kitchen printer.

In this case, the business dimension of the demonstration was covered through the live scenario which showed the software's capability to solve the client's issues. This helped to convince the client about the software product. Meanwhile, initial requirements were also collected. In this case, the client had two different types of menu: one for local customers and another for tourists. The client also had three types of service: takeaway, delivery, and internet service. After the live scenario demonstration the client asked for each different kind of service to have its own printer in the kitchen site, so that when the PDA was used it would not send all orders to the same printer.

In summary, it was clearly beneficial to plan the software demonstration by focusing on the client's specific issues and to develop a live scenario, as this best showed that the analysts had a possible solution to the client's business process needs. It should be noted that in this case, the client accepted the software offer after the software demonstration.

5.3.3. Time Constraints and making 'good enough' demonstrations

> *The flexibility that we want to have during software demonstration is constrained by a time limit since we only have one hour and a half to present our software….so we have to do our best to explain our software functions to the client…..Our strategy is not to make the client stay for a long time in the software demonstration….we do not want to make the client feel tired and we do not want to spend a lot of time explaining software that really needs months to explain; rather, we focus on the main issue and provide a live scenario through which the software can demonstrate a possible solution….[Team Leader]*

From the above comments it can be observed that constraints on the time allowed for software demonstrations made analysts concentrate on very specific client issues during demonstrations. In response to such limitations, analysts created strategies that related closely to the client's business processes, and in which they tried to minimise coverage of unnecessary information in order to help the client feel comfortable. Of note is that the analysts focused on the business dimension more than on the software analysis dimension. This is understandable given that the software offer had not yet been accepted by the client.

5.4. Mechanisms of Scoping and Creating a Packaged Software Offer

The scoping process involves software analysis through the discussion of high level modification requirements and new features. Without scoping, the software implementation time frame might expand to the degree where this would impact negatively on the cost and time involved in PS implementation. A participant explained:

> *It was a big issue, and it had required a lot of effort and time to work with that undefined scope to understand business practices and business requirements. Therefore, our strategy now is to define the scope of the software early on so that we will be prepared for the next step in the case that clients accept the software offer [Team Leader]*

For example, a team of analysts at one of the participant organisations demonstrated a Human Resource Management System (HRMS) to their clients. HRMS deals with demographic data, current employment information, employment history, qualification tracking, and salary information. In this case, the company's general manager also asked the team of analysts to visit the client. The analysts' team and I went to meet the client to discuss her needs. The client already had HR software and was experienced in its use. In this case, the analysts started by asking her about the issues that she and her employees faced with the current software. We discussed the client's issues and the possible solutions. As one participant explained:

> *If the client already has software and he /she needs to replace it, there will definitely be some functions that our current software does not provide, so we need to know what these functions are and then assess whether we can provide them, otherwise the client will not buy our software since we could not provide solutions for his/her issues[Team Leader]*

After the visit, the analysts' team provided an assessment report to the general manager. This report included a discussion of the client's issues, the modifications required, and new features to be added. The client's issues involved were categorised into various types of issues: transaction issues, such as the employees' bonuses mechanism and costs related to the provision of uniforms, and output format issues, such as the software reports format. The general management and the analysts' team met to discuss possible ways of modifying the packaged software in order to fit with the initial requirements, and to discuss the analysts' expectations of the development time-frame involved. After this, they also resolved client issues related to the price of the software (this is to make the software offer). Hence it is clear that one part of creating a software offer is scoping through software analysis. During this process of creating a packaged software offer, core requirements are emphasised but detailed requirements are neglected. This leads to a further question: how do the packaged software companies manage requests for new requirements and features after the packaged software offer has been accepted?

5.4.1. Packaged Software Offer Elements

> *[Team Leader] has defined the following elements for a packaged software offer:*
> *Initial requirements which consist of core requirements. In other words, high level modification requirements and new features that are related to transaction functions in packaged software. For example, HR software that includes transactions requires such things as an employees' bonuses mechanism and uniforms. In H2O, client has two different types of menu item, one for local customer and other for tourism; they also have three types of service: takeaway, delivery, and internet service. Client asked for each type of service to have its own printer at kitchen site so not all PDAs send an order to the same printer.*

Software output forms which allow a number of modifications by the client on packaged software, such as changing a report format output.
Technical dimension that relates to the client infrastructure site such as server, software program and so on. For example, in the case of H2O the software offer included consideration of hardware requirements needed in order to run the software [Researcher's field notes]

It can be seen that when creating a packaged software offer, the software companies consider the scope of the offer according to the initial requirements, the number of modifications requested by the client, the nature and extent of the modifications, and the technical requirements involved. The software company gives clear guidelines regarding which pre-conditions have to be fulfilled for a packaged software implementation to go ahead. By focusing on such elements of a packaged software offer, the software company was able to work towards its main goals of accurately estimating the cost and time required for packaged software implementation. Such estimations are generally based on the extent of the differences between what the packaged software functionality offers and the client's business process. From my observation, analysts' estimates of the cost and time required could only be informed guesses based on their prior experiences of working with modifications.

After we understand the client's needs – whether they want transaction function modifications that require 'customisation,' or new features - we will assess the cost and time required to develop such requirements [Team Leader]

As seen below, when creating a packaged software offer, the software companies also consider issues related to various kinds of 'assessment criteria'. The different kinds of assessment criteria are related to differing kinds of offer elements, and are used to measure the level of impact on effort needed to develop, customise, and modify the packaged software. When we outline the relationship between the type of elements and the assessment criteria, this type of relation also highlights the software company's duties and the client's duties.

[General Manager] has defined the following assessment criteria for packaged software offer:
A New Features request that consists of developing a new function that changes the existing package
Customisation which consists of modifying the existing functions to fill the gaps between the functions offered by a software package and the client's needs
Software Output which consists of creating new reports or modifying existing reports or modifying existing screen layout, input data or output data
Technical needs which consist of assessing the client's infrastructure requirements such as hardware and software

5.4.2. Discussion of Packaged Software Offer with Client

> *It is very important that the correct information is gathered at this stage, if we are to implement the software successfully. So we meet with the client to explain our offer and discuss the offer information and see if the offer information can already meet their need. We also estimate the cost and the time involved with such an implementation [Team Leader]*

After the packaged software offer has been created by the software company according to the guidelines established by the elements and assessment criteria explained above, analysts and sales teams meet with the client to discuss the offer. This process has two purposes. Its business dimension consists of convincing the client to purchase the software and includes discussion of the package's price. Meanwhile, issues related to the software analysis dimension include the following software offer elements: initial requirements, technical dimension, and output form.

Once all the elements are finalised during the creation of the packaged software offer and the offer is accepted by the client, the manager can inform the team leaders that the offer has been accepted. However, for the team leaders' staff members, the work is only just beginning.

5.5. Pre-implementation Feasibility study

In the ethnographic account in the first part of this chapter, I have described some of the day-to-day working practices of an analyst, which are almost exclusively connected to working with the clients via software demonstration and creating a packaged software offer. The pre-implementation practices stage in this study resembles such feasibility studies as those used in RE practices at a high abstract level. This is because feasibility studies in RE and the packaged software pre-implementation stage discussed here are similar in terms of their purpose, such as dealing with software objectives, time and budget. However, at the practical level, pre-implementation practice has its own specification. Table 5.5-1 uses a feasibility study in RE (Sommerville, 2004) to assess pre-implementation practices.

Table 5.5-1 Assessment of Pre-Implementation

Elements	Feasibility study	Pre-Implementation Feasibility study
Goals	Are the overall objectives of the organisation satisfied by the proposed system? Can the system be developed with the proposed budget and timeline?	What are the client issues? What is the possible solution? Is the possible solution within the scope of the software company's domain? What are the cost and time required for a possible solution?
Business dimension	Worthiness of proposed system.	Instilling confidence in the client, securing business, and creating a software offer.
Software analysis dimension	Information gathering to assist the assessment of proposed system.	Information gathering to identify client's issues, new requirements and new features needed (if any) to assess cost and time for proposed solution implementation.
Stakeholders	Management of departments, experts, technical professionals, and people who are familiar with such a system.	Potential client, client's issues, client analysis information and client company structure information
Tools, methods	Interviews, questionnaire.	Live Scenario, and discussion and negotiation.
Domain knowledge	The development organisation and the customer can cooperate to ensure that the domain is understood.	The development organisation has to be an expert in the domain.
Assessment criteria	Objectives of the organisation are satisfied by proposed system. System is developed with the proposed budget and timeline.	A New Features request, Customisation, Software Output Customisation, and Technical Needs
Critical Decision	Considers the worthiness of the proposed system, or regards changes, development decisions, seclude and budget.	The possible solution is within the software company's domain.
Output	Feasibility study report and recommendations.	Packaged software offer, assessment report, and client issues, organisation structure and analysis.
Scoping Factors	Budget, timeline, technical and development issues.	Packaged software assessment criteria, elements, and limitation of work domain, client organisation size, and client's issues.

As can be seen from information provided in Table 5.5-1, there are a number of differences in practices and purpose between the elements of feasibility studies carried out in RE and for pre-implementation. For example, in pre-implementation of packaged software, the analyst must instead think about what the client's specific issues are and identify whether any existing packages offered by the analysts' company can offer a solution. The analysts engaging in pre-implementation must also consider the possibility of refusing a request for a particular solution if that solution falls outside the scope of the company or outside the scope of the company's current products. The process of identifying whether the solution is within the company's scope may involve thinking about the time and cost involved with implementing a particular package or with making requested changes to that package.

The main goal of the 'business dimension' is that it engages with actually selling the proposed packaged system to the client by showing them how the package operates and how it could fulfil their requirements. The analyst carrying out pre-implementation must actively instil confidence in the client, secure his/her company's business, and create a software product offer.

When it comes to the software analysis dimension, analysts in both the general form of requirements engineering outlined by Sommerville (2004) and in pre-implementation carry out a range of activities to discover the client's issues that need solving and that help them to find initial requirements. They will later need to follow up on such requirements by checking whether new requirements are needed or new features need to be added to the proposed solution. If new features are required, they will again need to assess the cost and time involved with such requirements. In pre-implementation, analysts need to consider the modifications to existing functions that have been requested by clients.

As shown in Table 5.5-1, with pre-implementation, the analyst's considerations will be somewhat broader than in the general RE methods outlined and recommended by Sommerville (2004), as they need to first identify potential clients, then gather as much information as possible about the potential clients, and then prepare to attract the clients by identifying their issues that need solving. This is done via the use of forms relating client analysis information and by using the analyst company's databases that contain information about potential clients' company structures.

One of the tools used in pre-implementation is that analysts engage in Live Scenarios to demonstrate a proposed solution or find out what the requirements for the solution are, and at the same time, sell the solution, by carrying out discussions and various forms of negotiation. The pre-implementation analyst is engaged in promoting a system that meets the client's requirements, and in demonstrating and selling that proposed solution.

The level of domain knowledge required for the analyst engaging in pre-implementation of packaged software may be higher than that usually expected of an analyst conducting RE because the client will expect the development organisation to already be an expert in the domain and to offer them the best possible solution or a range of viable solutions.

The assessments criteria used to implement and modify the system in pre-implementation for packaged software involves its own set of assessment criteria. As detailed in discussions earlier, these assessment criteria involve a 'New Feature requested' criterion which assesses proposed changes to the existing package, a 'Customisation' criterion which assesses the impact that may result from modifying existing functions to fill gaps in requirements, and a 'Software Output/Input customisation' criterion which consists of creating new reports or modifying existing reports. The analyst engaging in pre-implementation will make a Critical Decision when deciding whether the solution needed by the potential client is within the domain of the analyst's company.

When it comes to the Output dimension of feasibility study, the analyst working with pre-implementation considers the project feasibility and responds by means of the assessment report, information gained about client issues, organisation structure and analysis, and the packaged software offer that is made to the client.

The last element of assessment is Scoping Factors. The scoping factors involved in pre-implementation practice for packaged software are influenced by assessment criteria, the packaged software offer elements, the limitation of the work domain, the client's organisation size, and the client's issues.

5.6. Summary of Pre-implementation PSIRE

The ethnographic account of the pre-implementation of packaged software at these companies presented within this chapter provides, for the first time, insights into how a software development company of this size (small-medium) approaches the challenge of managing the pre-implementation process at this point of the packaged software implementation life cycle. I have demonstrated the challenges associated with the search for potential clients and the challenges associated with software demonstration. I have also highlighted elements and assessment criteria involved with creating a packaged software offer that is based on modification and customisation of existing packaged software.

In this chapter, I have outlined the pre-implementation day to day working life of analysts. It appears that the mechanisms of the software demonstration request and the analysts' response facilitate the type of action in which both the software demonstration request and the analysts' response are impacted by the client's organisation size and the issues faced by the client's organisation. Meanwhile, the role of the analyst engaged in packaged software implementation involves analysis, marketing, and sales. In pre-implementation, the analyst is likely to be the staff member delivering the software demonstration. Because analysts know how the software is built, how it works, and how to explain it, clients may be more likely to buy the software when the analyst explains it to them. Clients will buy the software if they feel that the software will help them to achieve their objectives and to solve their problem. Therefore, the analysts' presentation of the software is approached from the perspective of convincing the client of the package's suitability and from the perspective of identifying misalignments. This presentation requires particular skills and knowledge related to packaged functions and how functions are related to each other, as well as communication skills and the ability to persuade the client.

Figure 5.6-1 shows the various mechanisms involved in pre-implementation as observed at the two case sites. While some processes occurring within this setting could have been uncovered through the means of reading the company documentation on pre-implementation processes, the dynamics of the companies that I have described in this ethnographic study and the information about their formal and informal approaches to pre-implementation provide a unique picture of the relationship between a packaged software company and its clients and reveals how the pre-implementation process works on a day to day basis.

Within the first part of this chapter, I have provided the first ethnographic narrative on pre-implementation of packaged software, drawing on data collected within two small – medium sized software development companies in the Middle East. In the following section, I re-address and analyse the ethnographic findings, focusing on the mechanisms of requirements engineering practices and analysts' roles and responsibilities.

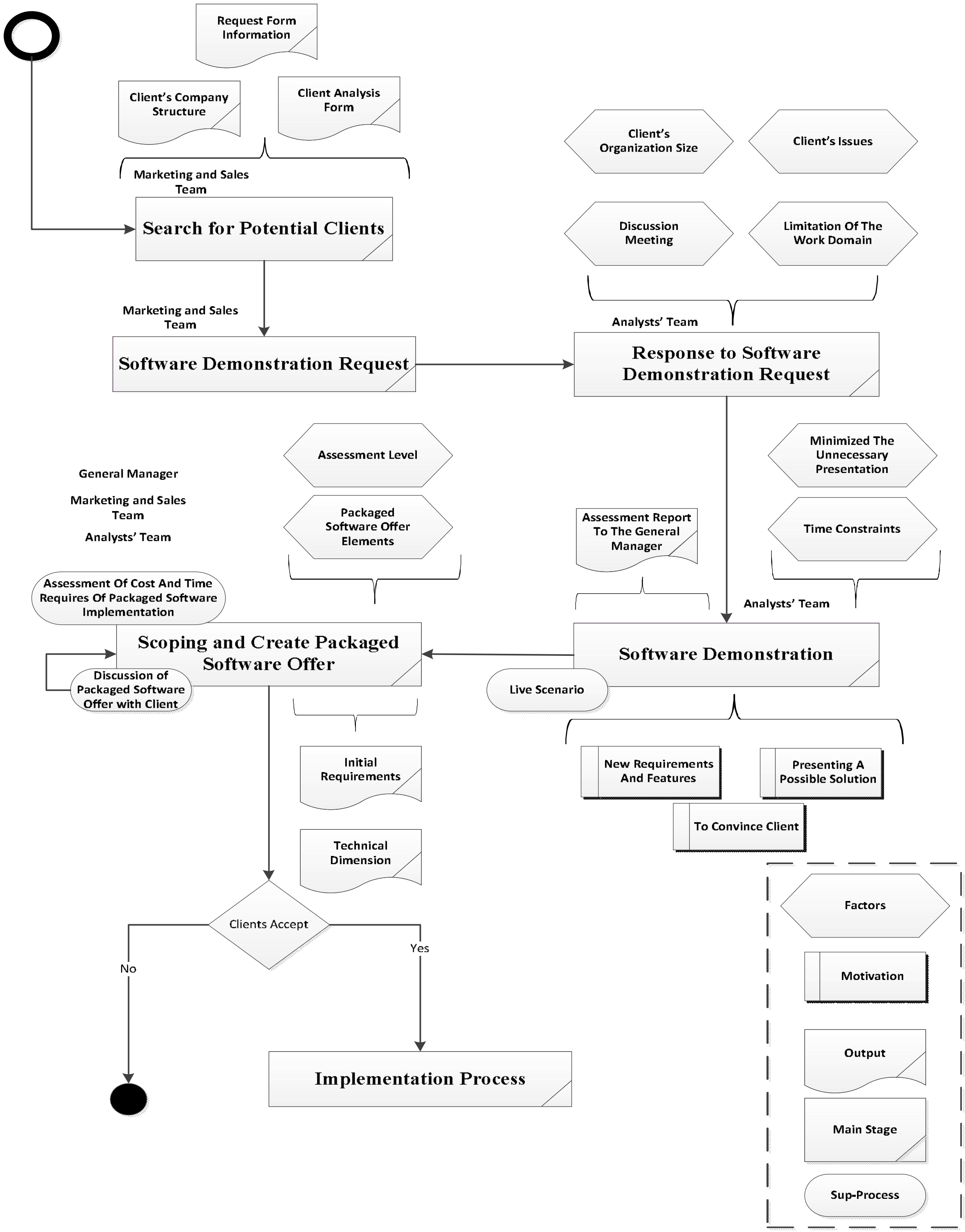

Request Form Information
Client's Company Structure
Client Analysis Form
Marketing and Sales Team
Search for Potential Clients
Client's Organization Size
Client's Issues
Discussion Meeting
Limitation Of The Work Domain
Analysts' Team
Marketing and Sales Team
Software Demonstration Request
Response to Software Demonstration Request
Assessment Level
Minimized The Unnecessary Presentation
General Manager
Marketing and Sales Team
Analysts' Team
Packaged Software Offer Elements
Assessment Report To The General Manager
Time Constraints
Assessment Of Cost And Time Requires Of Packaged Software Implementation
Analysts' Team
Scoping and Create Packaged Software Offer
Software Demonstration
Discussion of Packaged Software Offer with Client
Live Scenario
Initial Requirements
New Requirements And Features
Presenting A Possible Solution
To Convince Client
Technical Dimension
Clients Accept
Yes
No
Implementation Process
Factors
Motivation
Output
Main Stage
Sup-Process

Figure 5.6-1 Pre-Implementation Stage

Section II: PS Implementation

In this section, I address elements of the implementation stage that include (1) the mechanisms used to identify misalignments, (2) the mechanisms of assessment of misalignments, and (3) responding to users' needs/misalignments.

PS implementation involves several activities such as customisation, installation, configuration and adaptation. In the PS acquisition context, "detailed analysis comes after purchase. It is only during installation that users become deeply involved for the first time in assessing how the software meets their needs" [Sawyer, 2001]

5.7. The Mechanisms Used to Identify Misalignments

The process of identifying misalignments consists of conducting discussions with users (clients) to determine what feature wants and needs they may have in relation to the software on offer. Analysts install a copy of the packaged software in order to identify technical requirements and misalignments between packaged software technical requirements and users' IT infrastructure. Analysts then use the installed copy of the software to provide a software demonstration to users. This helps the analysts to identify the business misalignments between the software functionalities and the users' business process functions, leading to customisation, new features, and modified output requirements.

After the software offer is accepted by the client, we use the information from pre-implementation stage to start collecting more details about the user's needs... So first, you install a copy of our software then we start to explain our software functions to determine mismatches between our software and the user's business process [Team Leader]

Two questions arise from these observations. How does the analysts' team approach identifying misalignments? What is the purpose of installing a copy of the packaged software? I discuss these questions in the following sections.

5.7.1. Why Analysts Install a Copy of the Packaged Software

In one of the cases I observed, a Human Resource Management System (HRMS) had been offered to an organisation. Since the client had accepted the software offer, further in-depth understanding of the users' needs was required. In this case, the analysts started to identify technical misalignments between packaged software technical requirements and the users' IT infrastructure via the installation of a copy of the packaged software. The analysts spoke about the installation of the copy as a way to discover the technical requirements for the software to be implemented. In other words, the analysts used this activity to determine if there would be any software integration issues or software infrastructure issues involved with a real implementation of the software. This also shows that when implementing packaged software, there is a great need for certainty regarding whether what the packaged software infrastructure requires and what the users' IT infrastructure delivers match each other. This issue could also come up in bespoke RE, but usually much later.

We cannot ask users about their infrastructure because most of the users are not IT people and even IT people don't know some of the infrastructure requirements for software to be run……We try to discover any issues with the users' infrastructure, but we do that by installing a copy of our software … that's the only way to get to know the issues with the users' infrastructure…[Team Leader]

In the case of this HRMS software, several technical issues were discovered by installing the copy of the software. For example, issues were found that related to server compatibility, such as speed, storage space, and RAM size. Other issues were found on the users' desktop side, such as their computer missing some components that were related to running files.

Everybody had to go and visit each of the desktops to install the apps... the users' computers are not compatible with our software requirements so we have to fix them[Team Leader]

Hence, it is clear that analysts need to identify the misalignments between software technical requirements and users' infrastructure capability in order for software to be implemented. Another question that arises is how is such a copy of the software used by analysts to identify business misalignments that may need to be addressed by the introduction of new features, customisation, and modifications to output?

5.7.2. Software Demonstration

Through their [users'] interactions with a package, users can more easily identify the functions they need and desire, and define the functions that are not available in the package [El Emam & Madhavji, 1995]

The analysts spoke about this installation of the copy as a way to educate users about the software's functionalities, to help users create a vision about the software functions, and to increase users' participation in discussions.

We start explaining the software functions. So we install a copy of the software, not to be used in a real process, but to help us to explain our software functions to users..... I think that helps the users know what software can do.... As you know, users sometimes have no understanding of what software can do, so we use a copy of our software to teach them, and to help users to engage in a good discussion about the software functions and their business process [Team Leader]

After the software was installed successfully, analysts used the version of the software to carry on identifying misalignments by following issues mentioned on the report from the pre-implementation stage. For example, in the case of this HRMS software, the users' issues were categorised under 'transaction issues' such as 'add employees', 'bonuses mechanism', 'payments made for uniforms', and 'output format issues' such as the software reports format. Analysts spoke about how using this installed copy of the software could minimise the customisation effort.

In this case relating to HRMS, a transaction misalignment was found, which required customisation of the software. Analysts explained to the users the functionality related to payments made for uniforms. The users accepted the interface layout and the output data but asked about the customisation of a relationship between 'payments made for uniforms' and 'employees' salary'. That is, the software needed to include a mechanism by which 80 JD was deducted from the first month of an employee's salary, as a guarantee for uniforms, and then returned to the employee after 3 months. In this case, the analysts minimised the customisation effort by explaining how the software could help users when kept in its present form, and then agreeing to customise the software in terms of a transaction formula. More comprehensive discussions of misalignment types can be found in Yen et al. (2011) and Sia & Soh (2007). However, the theories of Yen et al. (2011) and Sia & Soh (2007) do not hold for the investigated small – medium sized software development companies, as their theories consider only the clients' or users' perspective, not the perspective of packaged software companies.

There was a strong consensus amongst analysts that they should consider carrying out software demonstrations for packaged software as a means of convincing users that there were alternative solutions to misalignments. The general recommendation from analysts who participated in this study was that software demonstrations of a trial version of the packaged software should be used as part of the implementation process to educate users about the software's functionalities, to increase users' participation in discussions, and to discover and discuss user needs and misalignments.

5.8. Responding to Users' Needs/Misalignments

In general terms, analysts respond to the discovery of a misalignment in one of two ways: either deciding that the user's company should adopt the packaged software as it is, making the most of the functionality it does offer – a decision that might require the company to change their business processes – or deciding that the packaged software needs modification in terms of customising a function or adding new functionality. However, various factors need to be taken into consideration before the analyst decides what action to take in response to finding a misalignment.

The analysts first need to determine whether the misalignment that has been discovered is in fact an 'actual' misalignment, or only a perceived one. A misalignment is actually real only when the software does not support such a transaction or does not support transaction formula as required by users. A misalignment might seem to exist in cases where the software functions actually support a particular desired transaction, but in a different order. The requirements engineering practice for the development of bespoke software does not involve such distinctions.

If the misalignment is found to be an actual misalignment, then various factors need to be considered in terms of whether the impact of those factors make it possible to fix the misalignment or not. For example, if the misalignment is genuine, analysts next need to think about the software scope and determine whether the misalignment is within the software scope or outside the software scope. They will also need to consider the size of the users' organisation as this may affect the organisation's ability to pay for such customisations. After carrying out such methodical assessments of misalignments, the analysts can choose a course of action.

Another issue that must be kept in mind, however, is any details or decisions pertaining to what the software development company's strategy is when dealing with user's needs and with misalignments. For example, during my case studies, I found that both of the software development companies I observed wished to minimise customisation of software as much as possible.

In the above section, I have discussed some primary factors relating to how analysts determine how to deal with misalignments. However, there is also a secondary factor that might be considered important when misalignments have been discovered. This relates to the fact that finding a misalignment is not necessarily negative. There may be some benefits that can be derived from identifying a misalignment. The presence of a misalignment may sometimes provide the opportunity for software developers to test aspects of their software or to improve their current packaged software.

5.8.1. Identifying 'actual' misalignments vs. 'perceived' misalignments

> *We have to reduce making changes [customisation/ new features] in our software. Cause it's not easy to do. It's not easy to throw away software that has been developed over several years. Yesterday we met with users for HR software, and they kept stressing that the software did not have this function or this function, and so on. Actually, our software does provide those functions, but not in the order they want, so in such a case we explain to users how such functions work and the order of the function process.... However, if the request from the user is really something that does not exist in the software, such as a transaction formula, and if it requires us to make new features, then we have to make them [Team Leader]*

As explained earlier, before assigning misalignments to a specific category such as new features, customisation, and output, it should be decided whether a particular misalignment actually exists or whether it only apparently exists: a misalignment is actually real only when the software does not support such a transaction or does not support transaction formula. A misalignment might be perceived to exist in cases where the software functions actually support a particular desired transaction, but in a different order. In this section, I shall explain the analysts' responses to some specific misalignments in detail.

In one case relating to HR software, the accounting manager asked the analysts to add some attributes into employees' salary reports. However, these attributes were represented by other reports, so the analysts explained that to the accounting manager. As a result, the accounting manager accepted the software report order as it was, without requesting any further changes. In the other case mentioned previously, analysts explained to the users the functionality related to payments made for uniforms in HR software. The users accepted the interface layout and the output data but asked about the customisation of a relationship between 'payments made for uniforms' and 'employees' salary' that would involve a transaction formula that was not supported by the software.

Both of the situations involve misalignments, but in the case where the accounting manager asked for attributes to be added to the salary reports, the misalignment can be categorised as a 'perceived' misalignment. That is, the misalignment was such that it could be 'worked around' by carrying out a process in a slightly different way than was initially desired (by finding the attributes in other reports). However, the misalignment that was found in relation to payments for uniforms that were to be deducted from employees' salaries can be categorised as an 'actual' misalignment rather than a perceived one because the misalignment needed to be responded to with customisation. The misalignment was such that the user's business process could not work unless a customisation was made.

We can therefore observe in packaged software implementation, analysts may use work-arounds, but this is in order to minimise customisation, rather than to reduce conflicts between requirements. More comprehensive discussions of 'actual' and 'perceived' misalignments can be found in Van Beijsterveld (2006). However, the theories of Van Beijsterveld (2006) do not hold when considering this process from the perspective of packaged software companies.

5.8.2. Minimising Customisation

To fill the gaps between the functions offered by a software package and users' needs, most organisations will customise software during its initial implementation (Newman & Zhao, 2008; Zach et al., 2012). However, there are repercussions of engaging in customisation: customisation is usually associated with increased costs and longer implementation time.

> *Customisation of packaged software is often not a trivial activity, as it can involve several hours – or even months – and can involve a substantial cost. It is therefore essential that companies provide value through their packaged software by placing an emphasis on recognizing their client's requirements of the software being implemented [Khoo et al., 2011]*

In the two software companies that I examined, the general managers and analysts supported the idea that users should adopt a package's software functions as they are and change their own business processes, rather than seek to modify the software to fit their particular business practices. One of the reasons for this recommendation is that, while the cost and time involved with customisation can vary, it often adds a great deal of additional time and money to an implementation effort. One other consideration is that the business processes of the user may be so complex that carrying out the modifications desired by the user might have significant impact on the software functions. Therefore, the particular issues that push users toward asking for customisations are to some degree in conflict with ideas sometimes expressed by the analysts' team leaders or by the analysts' general manager. The General Manager of one of the companies stated that

> *Users should try to adopt the processes and options built into our software, rather than seek to modify the software to fit their particular business practices… customisation needs a lot of time and costs a lot of money…Sometimes the user's business process is unbelievable… it is very complicated and needs a lot of time to understand and then a lot of time for us to customise our software [General Manager]*

Part of the purpose of the mechanism of minimising customisation and dealing with misalignments is to ensure the analysts' better understanding of what is redundant in software and what functions are essential for the operation of the software. This process can also involve identifying which customisation requests can be met without disrupting the software. Such considerations must extend to involve users' needs, the scope of the project, and customisation risk. In PS implementation, when users inform the analyst that a particular function is redundant, the analyst has to consider whether the unwanted function is actually connected to other functions of the software. As discussed in the Team Leader's comment below, if changes are made that affect the redundant function, this may impact other areas of the software. Therefore, such considerations also need to be kept in mind as part of misalignments assessment, as the analyst considers customisation decisions. In the case that the redundant function cannot be deleted or switched off, the user needs to adopt the functions of the software, even if it does not match with the user's business processes. This could lead to changes and adaptations to the user's business process structure.

> *Look ... first of all we try to minimise any change [customisation] on our software, but in the case that a misfit is found, we evaluate it based on the scope (whether it is within our scope, or not), then we evaluate it based on change [customisation] risk.... So that depends on the user's request, for example, if he/she wants new features (input, output and code[transaction]) or customisation of existing functions (input, output, or code [transaction]). We can't just accept every single request from the users..... There are so many issues involved for software to be implemented. The thing is it is not simple as.... We cannot just delete or change what users want from our software... in some cases these functions are closely related to each other, so deleting or changing one of them could impact on another one [Team Leader]*

During this ethnographic study, I observed that even though analysts try to encourage users to adopt the existing functions of the software package, when there are actual misalignments analysts are left with no choice but to agree to customisation.

> *As you know, some users ask us to change our software functions because perhaps their transaction formula is different from our formula. Even though we thought our software supports the kind of transaction they want, it still uses a different formula than theirs ... this requires us to change the software code..... [Team Leader]*

Although the two software companies I studied therefore sometimes carried out software customisation, the general stance expressed by the analysts I interacted with was that "users should try to adopt the processes and options built into our software".

5.8.3. Software Scope

As discussed previously in section 5.3 of this chapter ('Mechanisms of Scoping and Creating a Packaged Software Offer'), 'software scope' refers to those elements of a software package and the software implementation that are within the range of any packaged software offer that was made by the analyst company and accepted by the client during the pre-implementation. When requests are made to analysts regarding various misalignments, the analysts can therefore refer back to the pre-implementation packaged software offer.

We can't just accept every single request from the users - in some cases the users' needs are outside our scope so we explain the extra cost to our clients or we just inform users that we cannot meet their request [Team Leader]

There are three options an analyst can choose from in response to a request for dealing with misalignments that are outside of the software scope. The first choice the analyst could make is to advise the user that the change is not possible and to encourage them to adapt their business practices to the functions offered by the software as it is. A second possibility for the analyst is that they may be able to show the user that the functions of the software on offer can actually already meet their need; the software simply meets this need in a way that is different than what the user expected. The analyst's last option is to assess the cost and the time involved with meeting such user needs.

In such an instance, where the users' needs are outside the scope of what the analysts and users had previously agreed upon in the software offer, the analysts estimate the cost of a customisation by looking at the extent of the misalignments and the customisation risk involved. Analysts then provide a report that explains the extra costs related to the users' needs. In most cases where users' needs were outside the scope of the software offered, users agreed to adapt their business process to match the software. I could not obtain more details from users to understand their perspective about this case. However, the analysts stated that it was likely the cost and time issues led the users to adopt the software functions even though they might affect their business process.

I think the client's budget in most cases leads them to decide not to add other needs outside the scope, so in most cases they just use our software functions as they are [Team Leader]

To summarise, user needs might lead to the development of new features, or to the customisation of existing functions. Both of these options mean that a package can be adjusted to fit the business needs of a company. However, this can be a complicated process, and adds further costs to implementation. Moreover, having modified a package can lead to difficulties in the feature if a company needs to upgrade their packaged software, as the upgrade will no longer mesh with particular modules of the package.

In cases where the users' additional needs are within the scope of the software, analysts will tend to ask for more information about the business's processes and activities. It is very important that the correct information is gathered at this stage if the analysts are to make valid estimations of time involved, and if they are to develop the customisation successfully. If the analysts realise they can satisfy the customisation request, the analysts provide a report estimating the time needed for the customisation effort and the customisation risk involved. This is provided to the users and to the head of management.

If users' needs are within our scope, we need to evaluate the time frame required for customisation. But if the customisation required is high risk we try to get users to adopt our software functions [Team Leader]

For example, in one of the companies studied, an issue appeared in relation to an HRMS system. Analysts explained the functionality related to the 'bonuses mechanism' that their software provided, but it was not suitable for the users' bonuses mechanism process. Therefore, analysts discussed the bonus mechanism process provided by the software and the analysts and users also had an in-depth discussion of the users' bonus mechanism. During this process, the analysts were able to identify misalignments. After the analysts collected details about the users' bonuses mechanism process, the analysts provided a report estimating the time required for any customisation effort.

5.8.4. Organisation Size and Price of Software

A range of other factors inform customisation decisions as well, and customisation engaged in by various companies varies. Some clients, especially small and medium sized companies (SMEs), choose to carry out a large degree of customisation. This is because they may view it as essential that they keep their own best practices that they believe give them a competitive advantage.

The strength of Small and Medium Sized Enterprises (SMEs) is having unique business processes, so adapting those unique business processes to the standardized packaged software could be fatal for SMEs [Quiescenti et al., 2006]

A larger SME may be able to afford a greater number or extent of customisations. The pre-implementation stage outcome, the users' organisation size, and the software price are all factors that impact on decisions to customise the software functions. As suggested in the example below, if an organisation can afford to pay for a particular customisation, it will likely go ahead.

With an open scope, especially with a big organisation, we feel like we develop the software from point zero.....There is a difference between a small organisation and a big organisation, small organisations most of the time pay less and you know, we just go through implementation and then training but sometimes we still face some challenges such as the users' IT infrastructure.... With a big organisation it is another story, one software implementation took two years to finish; their business process was very difficult. But we have to do it... Very simple rules here, if the client wants his/her business strategies to be applied by the software and pays us a lot of money they need everything to be like their existing process... It is all about how much is paid...[Team Leader]

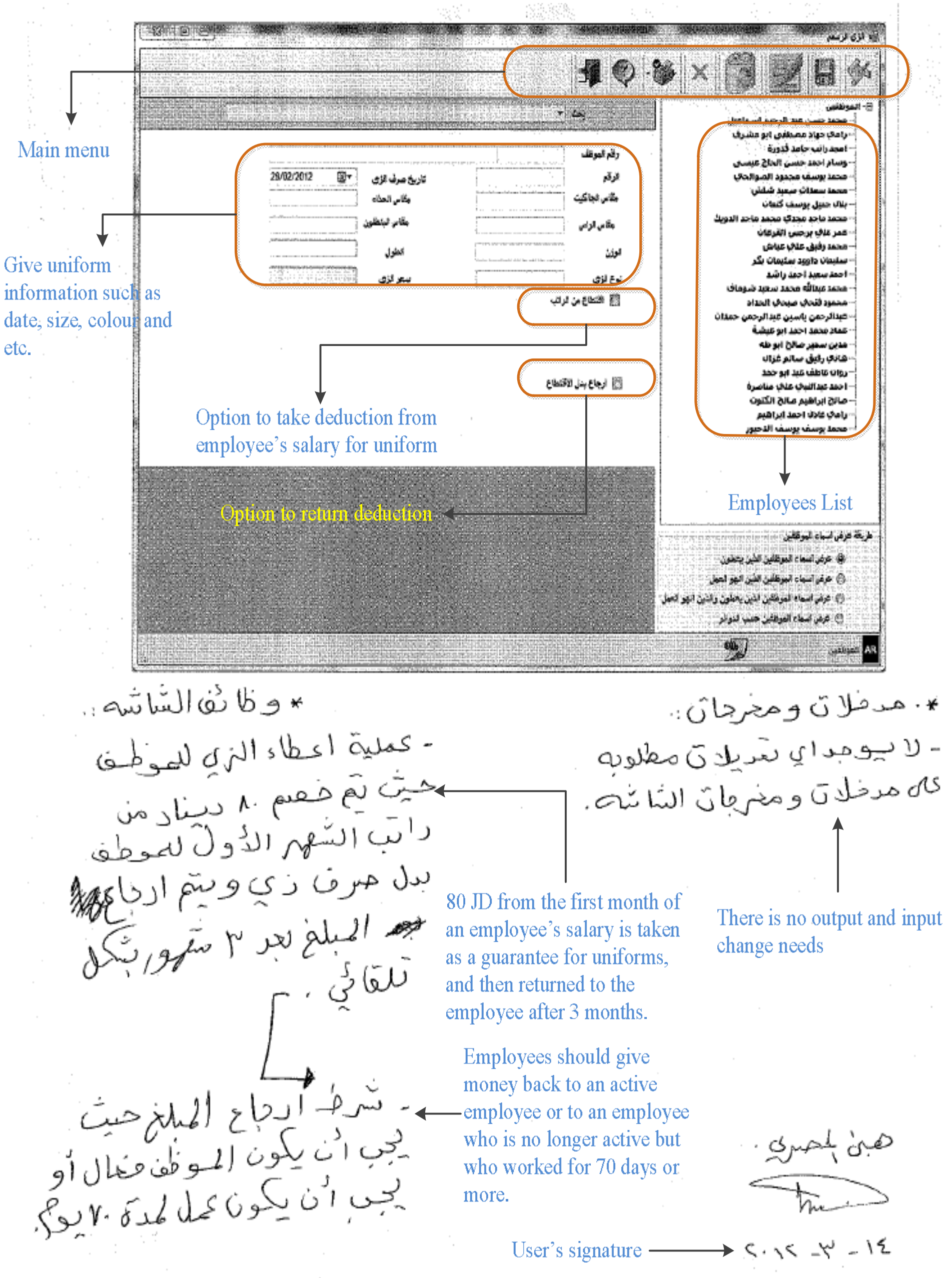
In summary, customisation decisions are usually associated with challenges involving increased costs and longer implementation periods. However, factors such as the existence of actual misalignments, the users' organisation size and the price that the client pays for the software have an influence on customisation decisions and sometimes leave analysts with no choice but to customise.

5.8.5. Benefits to Software Companies Derived from Users' Needs/Misalignments

Some users ask us to change our software functions or to add new features. Anyway, in this case some of these requests could be useful for our software so that we can improve the software functionality, so we do it even if it's outside our scope. But still, assessment of customisation risk is important, so maybe we can't add the function to the package that the current user wants because its development needs a lot of time, would cost a lot to do, and is outside the scope. But if we consider these features as important to add to our software, maybe these features will be in our next release [Team Leader]

As suggested in the comment made above by the analysts' team leader, a software development company may sometimes benefit from the identification of misalignments. For example, when a user points out functionalities that the package does not provide and that they desire, even if the software development company cannot develop that function in time to include it in the current package or considers it too risky or costly to include in the current package, the software development company may still have been provided with an idea for useful functionality to add to a future version of the package. There may therefore be some unforeseen benefits that arise for the software development company from discovering misalignments. Even if the company cannot meet the client's demand regarding the functionality at the particular time it is requested, the company may still have been provided with a future idea for the enhancement of their software package or may be able to promise the client that they will add that new function in their next release.

5.9. User Needs/Misalignments Validation Strategy



الزي الرسمي

بحث

رقم الموظف			
الرقم		تاريخ صرف الزي	28/02/2012
مقاس الجاكيت		مقاس الحذاء	
مقاس الرأس		مقاس البنطلون	
الوزن		الطول	
نوع الزي		سعر الزي	

☐ اقتطاع من الراتب

☐ ارجاع بدل الاقتطاع

الموظفين
- محمد حسن عبد الرحيم اسماعيل
- رامي جواد مصطفى ابو مشرف
- امجد رائب حامد قدورة
- وسام احمد حسن الحاج عيسى
- محمد يوسف محجوب الصوالحي
- محمد سعدات سعيد شلش
- بلال جميل يوسف كنعان
- محمد ماجد مجدي محمد ماجد الدويك
- عمر علي برجس القرقان
- محمد رفيق علي عياش
- سليمان داوود سليمان بكر
- احمد سعيد احمد راشد
- محمد عبدالله محمد سعيد شوملي
- محمود فتحي صبحي الحداد
- عبدالرحمن ياسين عبدالرحمن حمدان
- عماد محمد احمد ابو عيشة
- مدين سمير صالح ابو طه
- هاني رفيق سالم غزال
- روان عاطف عبد ابو حمد
- احمد عبدالنبي علي مناصرة
- صالح ابراهيم صالح الكتوت
- رامي عادل احمد ابراهيم
- محمد يوسف يوسف الدحيور

طريقة عرض اسماء الموظفين
- ● عرض اسماء الموظفين الذين يعملون
- ○ عرض اسماء الموظفين الذين انهو العمل
- ○ عرض اسماء الموظفين الذين يعملون والذين انهو العمل
- ○ عرض اسماء الموظفين حسب الدوائر

AR الموظفين

Main menu

Give uniform information such as date, size, colour and etc.

Option to take deduction from employee's salary for uniform

Option to return deduction

Employees List

*. مدخلات ومخرجات:

- لا يوجد اي تعديلات مطلوبه
على مدخلات ومخرجات الشاشه.

There is no output and input change needs

* وظائف الشاشه:

- عملية اعطاء الزي للموظف
حيث يتم خصم ٨٠ دينار من
راتب الشهر الأول للموظف
بدل صرف زي ويتم ارجاع
المبلغ بعد ٣ شهور بشكل
تلقائي.

80 JD from the first month of an employee's salary is taken as a guarantee for uniforms, and then returned to the employee after 3 months.

- شرط ارجاع المبلغ حيث
يجب أن يكون الموظف فعال أو
يجب أن يكون عمل لمدة ٧٠ يوم.

Employees should give money back to an active employee or to an employee who is no longer active but who worked for 70 days or more.

هبة الحصري.

User's signature

١٤ - ٣ - ٢٠١٢

Figure 5.9-1 User's needs validation strategy

The example above in Figure 5.9-1 above shows the strategy that analysts follow in term of users' needs validation. I have added some descriptions in English to the example above because the comments were in the Arabic language. This example shows a function that relates to payments made for uniforms and to employees' salaries, in which the software function allows the users to provide uniforms for employees but to collect a guarantee payment for the uniforms that can then be returned to the employees at a later date. The function therefore includes an option to take a deduction from an employee's salary and an option to return the deducted amount.

As we see on the right button within the example, analysts stated that "there is no output and input change needs". On the other hand, if we look at the left button it is clear that users asked to change the transaction formula relating to providing a uniform for employees but charging them for the uniform. The new transaction formula that was desired was that 80 JD from the first month of an employee's salary is taken as a guarantee for uniforms, and then returned to the employee after 3 months. In this example, analysts wrote this new transaction formula on a print-out of a screen shot of the software interface, in order to validate it with users. In this case, once analysts discussed such needs with users, more description of users' needs was gathered. As we see, the users in fact then added a further condition to providing uniforms for employees. This further condition was that the employers should give deducted money back to active employees (after 3 months of work) or in the case that the employee was no longer active, the employee had previously worked for the company for 70 days. After the validation of users' needs, analysts asked the users to sign on the screen printout to show their acceptance.

Hence, using the printouts of the software function screens to add users' needs is a strategy that analysts utilise during the requirements validation practice for packaged software implementation.

5.10. Misalignments Specification Form

The example below shows the content of a misalignments specification form that analysts sent to developers within their software company, and represents a request from Client A to the analyst company. Instances of responding to users' needs and requests such as this frame the day-to-day work of analysts.

Client Name: A
Address: AAA
Time and Date:
Request type: Transaction formula customisation on uniforms function
Description: 'payments made for uniforms' and 'employees' salary'. The users wish to have the following formula built into the system: 80 JD from the first month of employees' salary is taken as a guarantee for uniforms, and then returned to the employee after 3 months.
Module Name: Payroll
Priority: High
Input: No new Field.
Transaction Formula: 80 JD is taken from the first month of an employees' salary; return the 80 JD after 3 months if the employee is still active or minimum working days is 70. The deduction of costs for uniforms should be saved in deduction table under uniforms deduction
Relation with other functions: Employees Salary, employees' status, salary report.
Database Tables: add uniforms deduction field to deduction table

In this case, the request is created as the analyst fills in a request form by using a software tool that was created internally by the analysts' company. This software tool is used by the analysts to exchange information within the analysts' company and to manage their work.

Even without having any prior knowledge of the analysts' organisation involved, one can gain insights about the analysts' organisation by making observations of the request form. The context and layout of the request form is very structured and the language used on the form is formal. The request form and the information encountered within it give it the impression of a technical document that is addressed to an audience familiar with this particular area and this software functionality.

The form also shows misalignments specifications and a request to add a field within the database. We can also observe that this misalignments specification form suggests the high degree of knowledge that the analysts have in relation to the package and how its software functions interrelate, as they had worked on its development.

We can also observe that the form allows for misalignments traceability. This ensures that each business need is linked to an actual requirement, and that each requirement is linked to a deliverable. All requirements are considered in relation to other requirements, to other solution components, and to other artefacts. This is good practice for the business analyst. The goal of the kind of tracing supported by this form is to ensure that requirements and solution components are always linked to a business objective. Traceability helps to ensure that every requirement has a defined business purpose.

5.11. Summary of PSIRE

The ethnographic analysis of the identification, specification, and validation of users' needs during packaged software implementation presented within this chapter provides, for the first time, insights into how an SPSV approaches the challenge of managing the packaged software implementation process at this point of the implementation life cycle. I have shown the challenges associated with searching for and identifying misalignments, dealing with such misalignments, and dealing with customisation and adaptation challenges.

As misalignments are identified, the varying criteria of seriousness of the misalignments, and other factors relating to organisation size, software price, and level of user resistance to the software within an organisation affect decisions about whether to customise software or whether to adopt the software as it is. The particular way that misalignments are assessed impacts on misalignment strategy decisions. From my observations, various analysts and their general manager supported the idea of reducing changes (customisation/ new features) to packaged software as much as possible. However, this opposition to changing the software can really only be limited to occasions of perceived misalignments, rather than those of actual misalignments. Meanwhile, by installing a copy of the packaged software and engaging in a software demonstration, the analysts may be able to reduce perceived misalignments and to discover and confirm actual misalignments. Figure 5.11-1 shows all of the mechanisms involved during the PS implementation stage.

This chapter draws on data collected within small – medium sized software companies in the Middle East. In the following chapter, I readdress and analyse these ethnographic findings, focusing on the analysts' perspective for result validation in terms of responsibilities during various implementation stages, and on the mechanisms of requirements engineering practices.

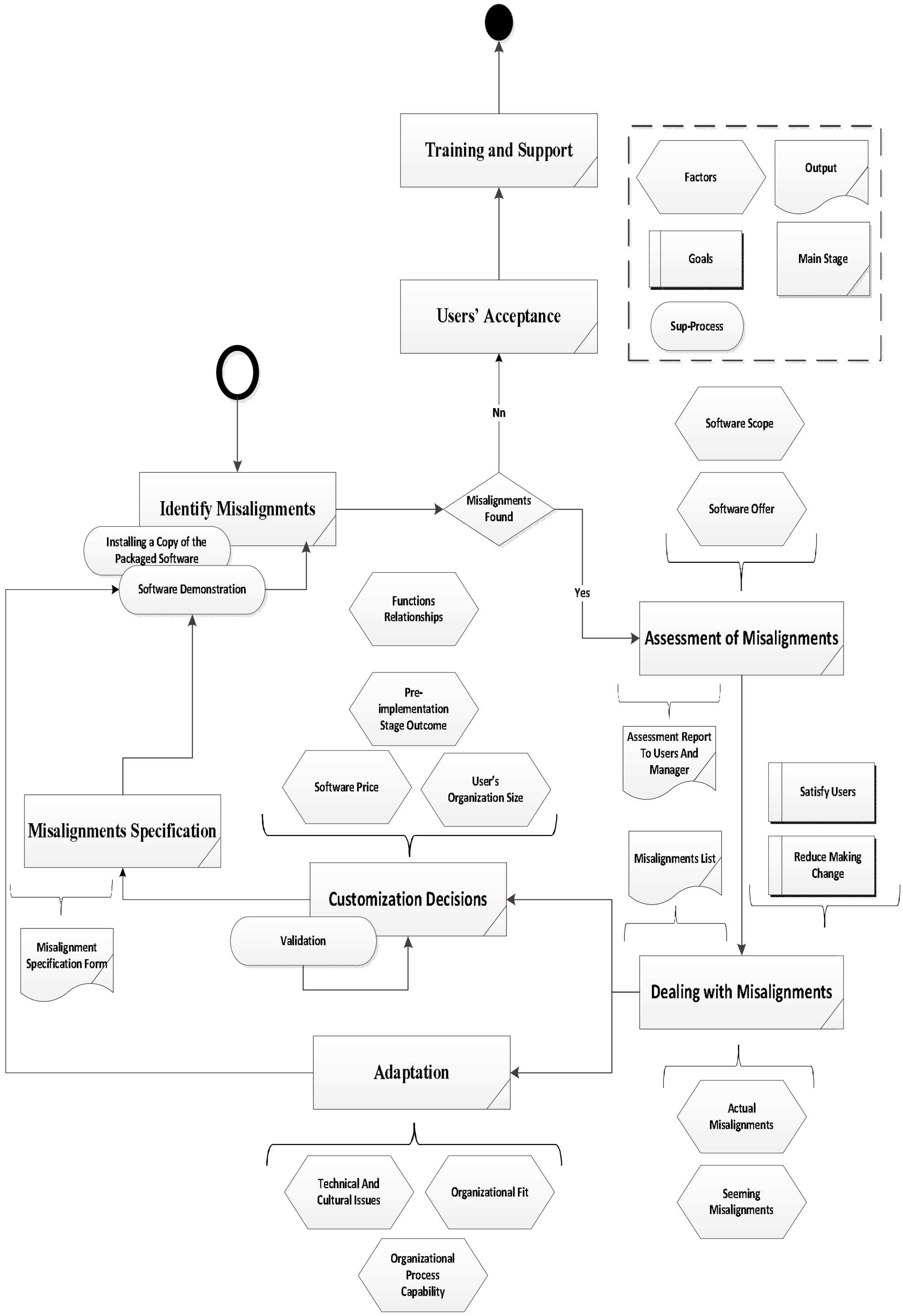


Figure 5.11-1 PSIRE Implementation

Chapter 6 Findings & Results: PSIRE Practices

6.1. Introduction

In the following sections I assess the enacted RE practices according to a framework adopted from Sommerville & Sawyer (1997) and Cox et al. (2009). The practices are described in the tables below in terms of levels to which they were used in the case organisations; the results are therefore based on the preceding ethnographic account. In this Chapter I also describe the validation of these results from the analysts' perspective. I use four levels of assessment of RE practices (as theorised by Sommerville & Sawyer (1997)). These levels of assessment are the following: standardised use, common use, discretionary use, and never used.

- Standardised use (SU): This practice has a documented standard and is always followed as part of the organisation's software development process i.e. it is mandatory.
- Common use (CU): This practice is widely followed in the organisation but is not mandatory.
- Discretionary use (DU): This practice is used at the discretion of individual project managers. Some may have introduced the practice for a particular project.
- Never used (NU): The practice is never or rarely applied.

The tables use guideline classifications relating to 'good requirements practices' that are suggested by Sommerville & Sawyer (1997) and Cox et al. (2009) to be 'basic', 'intermediate', or 'advanced'. 'Basic' practices can continually be repeated, and it is possible to estimate costs, time, and resources associated with these practices. In my assessment of the practices involved in PSIRE, 'basic' practices match with 'standardised' use. Meanwhile, 'intermediate' practices are more complex and lead to a 'defined' requirements engineering process. They can be considered as aligning with 'common' use. Lastly, 'advanced' practices are designed to help support continuous improvement within any RE process. Some of these practices involve advanced technology and advanced methods which require specialist knowledge. They may also involve expectations of and guidelines for organisational change. In my assessment of PSIRE practices, ‘advanced’ practices align with ‘discretionary’ use.

The results discussed in the following sections (6.2 through to 6.8) were obtained by my summarizing and synthesizing the findings I derived from the data collected during Focus Groups held with analysts from both Organisation One and Organisation Two, and from my day by day descriptive analysis of analysts’ practices in those organisations. During the Focus Groups I aimed to collect information from the analysts regarding how they carried out various RE practices when implementing packaged software. During the Focus Groups I asked, in turn, about requirements documentation practices, requirements elicitation practices, requirements analysis and negotiation, practices involved with describing requirements, practices involved with system modelling, requirements validation practices, and requirements management practices. The following sections contain (in italics) some reproductions of my summarizing findings (via descriptive analysis) determined as a result of the Focus Groups and written up at the end of each day, and from my day by day descriptive analysis of analysts’ practices. Note that the results obtained from the two different organisations were combined due to their close correspondence (see Appendix D).

I obtained my results relating to specific RE practices by discussing with analysts a list of practices adopted from Sommerville & Sawyer (1997) and Cox et al. (2009) and asking them to rate their use of each according to the four levels of assessment derived from Sommerville & Sawyer (1997). This helped me to understand the level of use of each named practice in PSIRE. This was understood by the analysts; however, I did run into the issue in each organisation that the analysts I spoke to did not always refer to the practices they used using the same terminology as myself. They therefore had issues with recognizing some of the names that I had supplied relating to RE practices. With each group, I therefore had to explain what was meant by each of the named practices on the list. Once this was done, with each group we proceeded to discuss each RE practice on the list. A total of 8 analysts (5 from Organisation One and 3 from Organisation Two) took part in these Focus Group discussions and the rating of the practices.

6.2. Requirements Documentation Practices

The requirements document itself is a document that effectively communicates requirements to customers, managers and developers. As can be seen from Table 6.2-1, the levels of all requirements documentation practices are considered 'basic' by Sommerville & Sawyer (1997) and Cox et al. (2009). When the use of these practices in PSIRE is investigated, it can be found that many of these practices receive Standardised use in PSIRE. Thus they are practiced at a very similar level to that suggested by Sommerville & Sawyer (1997) and Cox et al. (2009).

Table 6.2-1 Requirements documentation in PSIRE

Requirements Documentation Practices			
No	**RE Practices**	**Type**	**PSIRE**
RD1	Define a standard document structure	Basic	Standardised use
RD2	Explain how to use the document	Basic	Common use
RD3	Include a summary of the requirements	Basic	Standardised use
RD4	Make a business case for the system	Basic	Standardised use
RD5	Define specialised terms	Basic	Discretionary use
RD6	Make document layout readable	Basic	Common use
RD7	Help readers find information	Basic	Common use
RD8	Make the document easy to change	Basic	Common use

The results show that the most common standardised requirements documentation practices are to define a standard document structure (RD1), to include a summary of the requirements (RD3), and to make a business case for a project (RD4). The practices of explaining how to use the document (RD2), making the document layout readable (RD6), helping readers find information (RD7), and making the document easy to change (RD8), can be considered as 'Common use' practices in PSIRE. This means that these practices are widely followed in the organisations but are not mandatory. I also found that PSIRE documentation practices approached defining specialised terms (RD5) with 'discretionary use'. These findings regarding requirements documentation practices were drawn primarily from the day by day descriptive analysis forms. For example, here is an excerpt from my descriptive notes:

*We first discussed 'define a standard document structure'. The analysts understood this to mean there is a standard form that they use in order to document users' needs and their statements regarding this practice confirmed that they use this practice a great deal – in fact, analysts from each organisation told me that it was always used in their organisation. This practice therefore was found to be a 'standardised practice' in PSIRE. One analyst [user4] told me they considered it highly important because "we use to exchange information within our company and to manage our work" **(RD1)**. Other practices that they said they used all the time/every time, thus in a standardised way, were 'include a summary of the requirements', and 'make a business case for the system'. My task at this point was not so much to ask analysts why they used practices the way they did, but simply to find out to what degree they used them; however, sometimes the analysts did comment in ways that provided reasons for the level of use. For example, they always used and included 'a summary of the requirements' in requirements documentation because this seemed like such a basic necessity for paperwork and recording the documents would serve as the basis for all further development of packaged software. One analyst [user5] told me "we always make a summary of the requirements because this helps us make sure that every business need we're thinking about is linked to an actual requirement. We also want all of those requirements to be linked to a deliverable" **(RD3)**. They also always made 'a business case for the system' because it was important to clarify what users' needs are and what the changes were that they asked for regarding the packaged software functionalities. User5 said that "we use a misalignments specification form to explain a business case that users ask us to change with a specific function so developers will know where the changes should be" **(RD4)**. The analysts said that not all of the practices on the list were things that they accorded standardised use. For example, most of the practices on the list for 'requirements documentation practices' only received 'common use' by them. Analysts said, for example, regarding 'explain how to use the document' that they did this quite of the misalignments specification form and software demonstration request form are very structured and the language used on the form is formal **(RD6)**. Helping readers find information on the documents was also only given common use, not a practice used all the time. Analysts said that they sometimes did this carefully but did not bother to do it all of the time because of time constraints and because it was not that difficult to look through and find information anyway **(RD7)**. They only did this practice for very big or complex jobs or for reports with many pages. It was therefore obviously something that they thought was helpful to do, but it wasn't essential that they do it. I found from what the analysts said that the practice 'make the document easy to change' also had common use. Most of them said they did this most of the time. They suggested that they usually did this "**(RD8)** in case we need to change what users have asked us to do. We have to make it so any other analysts who have the permission to modify the users' request can do it in an easy way" [User3]. However, they didn't always do it because "normally only one or two of us deal with each new piece of software, so we already know what the users' needs are and we don't need any other analysts to make any changes". **[April 5, 7 2012 - Organisation 1 & 2 - Requirements Documentation Practices]***

Table 6.2-2 New Requirements documentation Practices - PSIRE

New Requirements Documentation Practices			
No	**RE Practices**	**Type**	**PSIRE**
RD9	Users' needs/Misalignments specification document	Basic	Standardised use
RD10	Estimating time needed for users' needs document	Basic	Standardised use
RD11	Estimating cost needed for users' needs document	Basic	Standardised use
RD12	Include users' needs validation document	Basic	Standardised use

During my field work, I also discovered a range of new practices that were carried out, that are related to requirements documentation. I have listed these 'New Requirements

Documentation Practices' in Table 6.2-2. The following excerpt illustrates the origin of two of these practices as captured in my analysis:

*What was an interesting result from these Focus Groups was that I found that there were actually a whole lot of requirements engineering practices related to requirements documentation that the analysts practiced and could tell me about that had not been included on my list. I realised that they were talking about new requirements documentation practices that were used only in PSIRE. The new practices they told me about were creating or including a 'users' needs/misalignment document', 'estimating time needed for a users' needs document', 'estimating cost needed for a users' needs document', and including a 'users' needs validation document'. It was interesting to hear about these new practices, especially to notice the fact that the analysts seemed very interested in first estimating the time and cost involved with creating users' needs documents. One analyst, User6, said that "we estimate the time and cost it could take to create a users' needs document because it helps us keep track of the resources we might need for a project. That can help us decide whether the project is worthwhile, and estimate the scope of the project before starting it" (**RD10** & **11**). It was also interesting to note that they sought later validation of the initial user needs and would document this validation. From what they said, they thought it important to check this to make sure they understood the client's needs properly and to keep a proper track of everything. User5 noted that "the validation of user needs can take place during our software demonstrations" **[April 5, 7 2012 - Organisation 1 & 2 - Requirements Documentation Practices]***

These new practices involve creating a users' needs/misalignments specification document (referred to as RD9) that analysts sent to developers within their software company. When analysts sent a specification document/request form to developers within their company, I was able to observe the following in my descriptive notes:

*The request is created ....as the analyst fills in a request form by using a software tool that was created internally by the analysts' company. This software tool is used by the analysts to exchange information within the analysts' company and to manage their work. Even without having any prior knowledge of the analysts' organisation involved, one can gain insights about the analysts' organisation by making observations of the request form (**RD9**). The context and layout of the request form is very structured and the language used on the form is formal. The request form and the information encountered within it give it the impression of a technical document that is addressed to an audience familiar with this particular area and this software functionality. The form also shows misalignments specifications and a request to add a field within the database. We can also observe that this misalignments specification form suggests the high degree of knowledge that the analysts have in relation to the package and how its software functions interrelate, as they had worked on its development (**RD9**). We can also observe that the form allows for misalignments traceability. This ensures that each business need is linked to an actual requirement, and that each requirement is linked to a deliverable. All requirements are considered in relation to other requirements, to other solution components, and to other artefacts. This is good practice for the business analyst. The goal of the kind of tracing supported by this form is to ensure that requirements and solution components are always linked to a business objective. Traceability helps to ensure that every requirement has a defined business purpose **[Feb 14, 2012 - Organisation 2 - Misalignments Specification Form]***

Other new practices that the analysts told me of are estimating the time related to creating the users' needs/misalignments document (RD10) and estimating the cost needed for creating a users' needs document (RD11). The scoping process carried out before beginning to develop any software involves software analysis through the discussion of high level modification requirements and new features. Elements of the scoping process are detailed in the following descriptive notes:

Today I learned about how the software analysts actually go about deciding how to create a software offer. I was present at the meeting at which they discussed creating an offer, and they explained to me what some of the most important factors to consider were. The reasons for meeting to discuss what elements should go into the software offer were that the analysts needed to define the software scope offer; the analysts needed to create and provide to the client clear guidelines regarding which pre-conditions had to be met for a packaged software implementation to go ahead; and the analysts needed to be able to work towards accurately estimating the time and cost required for the software implementation ***(RD10 & 11) [Feb 9, 2012/ Organisation 2 - Software Offer]***

Including a users' needs validation document (RD12) was another new practice that was identified. Analysts told me that using the printouts of the software function screens to add users' needs is a strategy that they utilise during the requirements validation practice for packaged software implementation.

All of these practices are a part of PSIRE, being practiced with 'standardised use'. Despite their being used so often and the fact that these practices are perceived as having a high value, these practices have not been identified during previous studies of RE and packaged software RE; this study therefore extends the current framework for requirements practices in term of PS implementation.

6.3. Requirements Elicitation Practices

Requirements elicitation is defined as a group of practices designed to help discover the requirements for a system. These practices are followed by analysts in order to elicit requirements from the stakeholders related to the system. However, the requirements elicited also depend on the application domain and on the organisational and operational environments of the system.

In Table 6.3-1 we see that, several RE elicitation practices are carried out at the 'basic' level; that is, they are almost always practiced. Practices RE1 through RE6 are practiced with 'standardised' use in PSIRE. However, just over half of the practices operate at the 'intermediate' or 'advanced' levels. Most of those practices that are 'basic' in RE are standard practices in PSIRE. Details capturing the use of such practices are provided in the excerpt below:

*During Focus Group discussions, analysts told me that they always assess the system feasibility (**RE1**) before taking on the client's project/developing any software offer. Methods they use to assess the system's feasibility conducted in the pre-implementation phase include the software offer, software scope and assessment criteria for packaged software offer. **RE2**, being sensitive to organisational and political considerations when eliciting requirements was something else that analysts from both organisations told me they do. Maintaining this sensitivity is possible by doing things such as "asking the right people about the requirements, respecting the client company's culture, and maintaining the level of formality the client wants" [user4] **[April 12 2012 - Organisation 1 & 2 - Requirements Elicitation Practices]***

Table 6.3-1 Requirements elicitation in PSIRE

Requirements Elicitation Practices			
No	**RE Practices**	**Type**	**PSIRE**
RE1	Assess system feasibility	Basic	Standardised use
RE2	Be sensitive to organisational and political consideration	Basic	Standardised use
RE3	Identify and consult system stakeholders	Basic	Common use
RE4	Record requirements sources	Basic	Common use
RE5	Define the system's operating environment	Basic	Common use
RE6	Use business concerns to drive requirements elicitation	Basic	Standardised use
RE7	Look for domain constraints	Intermediate	Discretionary use
RE8	Record requirements rationale	Intermediate	Common use
RE9	Collect requirements from multiple viewpoints	Intermediate	Discretionary use
RE10	Prototype poorly understood requirements	Intermediate	Standardised use
RE11	Use scenarios to elicit requirements	Intermediate	Standardised use
RE12	Define operational processes	Intermediate	Discretionary use
RE13	Reuse requirements	Advanced	Standardised use

In the following descriptive field notes one can see the analysts carrying out practices RE5, RE6, and RE11.

*I attended a software demonstration meeting during which the analysts aimed to show their product to potential clients and to convince them that the software product they had to offer would satisfy the client's needs. "We represent our software to a client by developing a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment. It really helps us to explain our software functions and connect these functions to a real case [Team Leader] (**RE5**, **RE6**)". The clients run a goods inventory and they currently have some software that they use in the company. However, their current software is not adequate for their needs and it currently has some issues. This causes problems like the staff of the inventory having problems following an order, missing orders, and difficulties relating to the inventory of items (**RE7**). All of these issues had been made known to the analysts through the sales team report (**RE1**). The analysts went into the meeting with the purpose of telling the clients about their own inventory software which they think will solve the client's issues*

*(**RE2**). The analysts did a few different things during this meeting. At the start of the meeting, first they clarified the client's issues. They kept referring back to the sales team report in order to ask about the client's issues (**RE1, RE2**). They then began explaining what their software could do to solve those issues faced by the client (**RE6, RE11**). They gave an initial description of their software product's functionality (**RE5**). It seemed that they did this in order to give the clients some kind of vision about the product's functions, and in order to increase the client's participation in the discussion **[Feb 1, 2012 - Organisation 1- Pre – Implementation/ Software demonstration]***

RE practices RE7 through RE13, shown in Table 6.3-1, are practiced rather differently in PSIRE than how they are presented by Sommerville & Sawyer (1997) and Cox et al. (2009). In general RE, a large range of practices could be considered as 'intermediate' practices, that is, they are more complex and not always practiced.

Those practices regarded as 'intermediate' include looking for domain constraints, recording the requirements rationale, collecting requirements from multiple viewpoints, prototyping poorly understood requirements, using scenarios to elicit requirements, and defining operational processes. In PSIRE, however, these practices are carried out at a range of levels. For example, prototyping poorly understood requirements and using scenarios to elicit requirements are carried out as standardised practices; this difference occurs because software is already developed and might not offer a perfect fit with user requirements. For example, in the following excerpt from my descriptive analysis, RE10 and RE11 are shown being carried out as a standardised practice by analysts engaged in PSIRE:

*While showing the client the software, the analysts also engaged in explaining different elements of the software. They explained its various functionalities and clarified various textual and graphical materials. The analysts tried to demonstrate the software in a way that showed how it could support the client's business (**RE11**). They also trained some users at the client's organisation on using some of the software functions. The analysts showed the client printouts of various software functions, in order to gather further client needs or gain more detail about the client needs they already knew about (**RE10**). They were therefore using a copy of the packaged software in order to explain the software functionalities (**RE10**), and in order to carry out requirements validation **[Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]***

Looking for domain constraints (RE7), collecting requirements from multiple viewpoints (RE9), and defining operational processes (RE12) are practices that are only carried out with discretionary use in PSIRE. We can see why looking for domain constraints would only be carried out with discretionary use in PSIRE when we consider that SPSVs generally work with SME clients. For the same reason, collecting requirements from multiple viewpoints also has discretionary use - if a software development company is working for an SME client, the number of users to be considered is small. One other requirements elicitation practice in this group, recording the requirements rationale (RE8), can be considered as having ‘common’ use in PSIRE. The excerpt from my descriptive notes below details my discussion of this practice:

*In our discussion about whether analysts doing RE for packaged software had to carefully record requirements rationale **(RE8)**, analysts told me that yes, this was a fairly commonly used practice. For example, User5 told me: “It’s best if we do record the requirements rationale because obviously it is smarter if we have a record of that. We might refer to the rationale to see if there is a real business need for a requirement or if it isn’t really needed. Or we might use this record to keep track of or stop scope creep” **[April 12 2012 - Organisation 1 & 2 - Requirements Elicitation Practices]***

The practice of reusing requirements (RE13) is an advanced practice, a practice used to improve a system, whereas in PSIRE, it has a completely standardised use. The reason that reusing requirements is so common in PSIRE is because the analysts and clients are almost always using and modifying pre-existing packaged software. In such cases, clients usually engage in as much reuse as possible so as to limit the number of customisations or other forms of modification. Another reason for such reuse is that the new software system or product will be partly intended to replicate or mimic the client’s old system. An example of reusing most of the requirements in a pre-existing software package is shown in the descriptive analysis excerpt below. Here, the client agrees that the package offered fulfils most of their needs, and they ask for only one new requirement:

*The analyst and client then discussed the information that the client had provided on the document forms. The analyst looked at the needs that the client had listed on their document forms, and explained how the current functions in the software provided the attributes that were needed **(RE13)**. If necessary attributes were missing, they wrote this down. The client decided to make a modification request regarding the attribute needed to collect family medical history, so modification request documents were filled out **[March 12, 2012 – Organisation 1 - During Implementation – Software demonstration & identify users’ needs]***

Table 6.3-2 New Requirements elicitation Practices - PSIRE

New Requirements Elicitation Practices			
No	**RE Practices**	**Type**	**PSIRE**
RE14	Use live software demonstration to elicit users’ needs	Basic	Standardised use
RE15	Use a user manual	Basic	Standardised use

I also identified some new requirements elicitation practices used during PSIRE. The new practices, listed in Table 6.3-2, are using a live software demonstration to elicit the users' needs (RE14), and using a user manual (RE15). According to Sommerville & Sawyer (1997) and Cox et al. (2009), these practices are carried out at the 'basic' level in general RE. My findings showed them to have 'standardised' use in PSIRE. It is interesting that the practice of holding live software demonstrations (RE14) is repeatedly used in PSIRE. The reason for frequent use of the live software demonstration in PSIRE is that it educates users about the software's functionalities, helps to increase users' participation in discussions, and helps analysts to discover and discuss user needs and any misalignments between these needs and what the software product or system can do (Sia & Soh, 2007). For example, here is an excerpt from my descriptive notes showing the use of a live software demonstration:

Today I accompanied the analysts on a trip to the client's company site. The analysts engaged in a software demonstration of the HR software they had developed for the client. The client had already accepted the software offer made by Organisation 2, but now a more in-depth understanding of the users' needs was required. I observed that there seemed to be various reasons behind the meeting/demonstration: some were to do with the software itself, some were to do with the client's requirements, and some were to do with the client's company environment. As the analysts showed the client the software and discussed the client's needs from the software, it was apparent that the analysts were trying to validate the client's needs (i.e. confirm those needs/requirements they had already collected), to get a better understanding of the client's business process, and to identify any mismatches between the client's needs and the software. These actions helped the analysts know more about the client's requirements ***[Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]***

In addition, a live software demonstration may help to convince users that there are alternative solutions to any misalignments that are identified (Al-Mashari & Al-Mudimigh, 2003): the live software demonstration can be used to show work-arounds (Khoo & Robey, 2011). In packaged software implementation, work-arounds are used with the intention of minimising customisation, not in order to reduce conflicts between requirements. In the software organisations I observed, 'work-arounds' were used when the analysts tried to convince clients to use the software as it already was, rather than ask for a full-scale customisation. This could be deemed a 'work-around' because the client does end up getting the function or result they want, just not in the particular order they desired, while the software organisation avoids having to customise a function. But this kind of work-around is only possible if the client's customisation request responds to what is only a 'perceived' misalignment rather than a 'real' one. If a required function is totally missing from the software or can't somehow be supplied by the existing software, then the function has to be built. Examples of the importance of these issues can be found in my descriptive analysis below:

> *The Team Leader told me that "We have to reduce making changes [customization/new features] in our software. Cause it's not easy to do. It's not easy to throw away software that has been developed over several years. Yesterday we met with users for HR software, and they kept stressing that the software did not have this function or this function, and so on. Actually, our software does provide those functions, but not in the order they want, so in such a case we explain to users how such functions work and the order of the function process ... However, if the request from the user is really something that does not exist in the software, such as a transaction formula, and if it requires us to make new features, then we have to make them". From what the Team Leader said today and from what I observed yesterday at the meeting he described, I have reached some conclusions about what issues are deemed most important by the analysts when deciding whether to fix misalignments or not (by making customisations or new features), and what sorts of guidelines they follow when making these decisions. I've observed that the analysts think about the following concerns and take the following actions: When analysts discover a misalignment, they respond in one of two ways: they either tell the user's company that they should accept the packaged software as it is – this might mean that the company actually has to change their own business process so that it aligns with what the software provides and how it provides it – or, the analysts agree to carry out the modification requested, either by customising a function or adding a new function. The decision of whether they should make the customization/ new feature or not seems to be guided by two considerations: first, the analysts decide whether the particular misalignment that was discovered is actually a real misalignment or not. I witnessed the analysts making a distinction between 'actual' misalignments and misalignments that were only 'perceived' by the client* ***[Feb 27, 2012 - Organisation 2 - Respond to the discovery of misalignments]***

It was found that using a user manual (RE15) is a standard practice in PSIRE. The purpose of user manuals in PSIRE is to educate users about the software's functionalities so that users will have initial knowledge of the software even before they use it. Analyst said that: "We always create a user manual for the client because we want them to know everything the software can do and so they will have a better appreciation of it. This might also decrease any confusion later on when they use it" [User2].

6.4. Requirements Analysis and Negotiation

Requirements analysis and negotiation are defined as practices that help analysts to identify and resolve problems associated with the elicited requirements. These may include identifying and resolving misunderstanding, incompatibility issues, and missing information.

Table 6.4-1 Requirements analysis and negotiation in PSIRE

Requirements Analysis and Negotiation			
No	**RE Practices**	**Type**	**PSIRE**
RA1	Define system boundaries	Basic	Standardised use
RA2	Use checklists for requirements analysis	Basic	Discretionary use
RA3	Provide software to support negotiations	Basic	Standardised use
RA4	Plan for conflicts and conflict resolution	Basic	Common use
RA5	Prioritise requirements	Basic	Discretionary use
RA6	Classify requirements using a multi-dimensional approach	Intermediate	Common use
RA7	Use interaction matrices to find conflicts and overlaps	Intermediate	Discretionary use
RA8	Assess requirements risks	Advanced	Standardised use

We can see in Table 6.4-1 that in requirements analysis and negotiation, many of the practices are considered basic elements of RE. For example, RA1 through RA5, which involve defining system boundaries, using checklists, providing software to support negotiations, planning in case of conflicts, and prioritising requirements, are all listed at the basic level in the table above. In PSIRE, RA1 (defining the system boundaries) and RA3 (providing software to support negotiations) have documented standards that are followed, and are always used. They can therefore be considered as receiving 'standardised' use. These findings regarding requirements analysis and negotiation practices were drawn primarily from the day by day descriptive analysis forms. For example, here is an excerpt from my descriptive notes relating to the defining of system boundaries:

Today I learned about how the software analysts actually go about deciding how to create a software offer. I was present at the meeting at which they discussed creating an offer, and they explained to me what some of the most important factors to consider were. The reasons for meeting to discuss what elements should go into the software offer ***(RA1)*** *were that the analysts needed to define the software scope offer; the analysts needed to create and provide to the client clear guidelines regarding which pre-conditions* ***(RA1)*** *had to be met for a packaged software implementation to go ahead;…….. One analyst told me that: "It was a big issue [scope], and it had required a lot of effort and time to work with that undefined scope to understand business practices and business requirements. Therefore, our strategy now is to define the scope of the software early on so that we will be prepared for the next step in the case that clients accept the software offer" [User5]. Another analyst said: "If the client already has software and he /she needs to replace it, there will definitely be some functions that our current software does not provide, so we need to know what these functions are and then assess whether we can provide them,*

otherwise the client will not buy our software since we could not provide solutions for his/her issues" [User6] ***[Feb 9, 2012 - Organisation 2 - Software Offer]***
So, I watched the analysts trying to install the software. The analysts' Team Leader said to me: "We start explaining the software functions ***(RA3)****. So we install a copy of the software, not to be used in a real process, but to help us to explain our software functions to users..... I think that helps the users know what software can do.... As you know, users sometimes have no understanding of what software can do, so we use a copy of our software to teach them, and to help users to engage in a good discussion about the software functions and their business process* ***(RA4)****[User4]"****[Feb 12, 2012 - Organisation 1 – Installation]***

Two practices RA2 (using checklists for requirements analysis) and RA5 (prioritising requirements) receive 'discretionary' use in PSIRE. This is understandable, since analysts, during packaged software implementation, use screenshots to validate user needs, rather than using a checklist. This is because the software has already been created. Also, during PSIRE, prioritising requirements is not a basic practice. Rather, analysts collect requirements in a circular process and develop those requirements that are agreed upon at the time or that their managers agree should receive priority (i.e. the use of this practice is at the managers' discretion). The practices RA6 and RA7 are used at the intermediate level in RE. However, in PSIRE, classifying requirements using a multi-dimensional approach (RA6) receives 'common' use (that is, it is used at a similar level as in general RE). Using interaction matrices to find overlaps or possible conflicts (RA7) does not receive common use or standardised use in PSIRE, however. In PSIRE, this practice is discretionary, not common or universal. Information supporting these conclusions was supplied from the following excerpt of my descriptive notes:

Our discussion of ***RA6****, classifying requirements using a multi-dimensional approach, led to my finding out that the analysts I questioned use this practice quite often, though not every time. The ways that they go about classifying requirements using a multi-dimension approach* ***(RA6)*** *include software demonstration, use print-out of a screen shot, and users' misalignments form. I asked why the considered this practice to be relatively important and why its use was common in their organisations. One analyst replied: "We consider it important because help us to reduce the misunderstanding between us the users, and also help us to pass the users' needs to developers in clear way" [User2], and another said "We have a multi-dimensional approach to classifying requirements because we could manage the users' needs and discuss it with them in different ways so we will reduce and conflict That's why we want to use it" [User6]. We also discussed* ***RA7****, 'using interaction matrices to find overlaps or possible conflicts'. The discussion revealed that this practice only has discretionary use in PSIRE: the analysts don't feel this is necessary very often and this practice is only followed if their manager feels there is a need for it. The analysts made some comments about the use of interaction matrices, saying that "they were of limited use when implementing too many modules of our software because we have to manage users' needs by consider how these needs related to software functions and other needs from different modules" [user4]. So they feel there is really not much need for doing this during the implementing of packaged software.* ***[Organisation 1 & 2 April 19, 21 2012 - Requirements Analysis and Negotiation]***

Lastly, 'assessing requirements risks' (RA8) is presented as an advanced practice by Sommerville & Sawyer (1997) and Cox et al. (2009), but is a standardised practice in PSIRE. The main activities that were a part of the analysts' assessments of requirements risks in PSIRE were thinking about whether customisations were really necessary or not, and whether going ahead with a customisation request could harm the software. Such considerations can be seen in the descriptive analysis notes provided below:

The Team Leader told me that "We have to reduce making changes [customisation/new features] in our software. Cause it's not easy to do. It's not easy to throw away software that has been developed over several years ***(RA8)****. Yesterday we met with users for HR software, and they kept stressing that the software did not have this function or this function, and so on. Actually, our software does provide those functions, but not in the order they want, so in such a case we explain to users how such functions work and the order of the function process ... However, if the request from the user is really something that does not exist in the software, such as a transaction formula, and if it requires us to make new features, then we have to make them* ***(RA8)****" [User5]. From what the Team Leader said today and from what I observed yesterday at the meeting he described, I have reached some conclusions about what issues are deemed most important by the analysts when deciding whether to fix misalignments or not (by making customisations or new features), and what sorts of guidelines they follow when making these decisions* ***(RA8) [Feb 27, 2012 - Organisation 2/ Respond to the discovery of a misalignment]***

Once more, I identified some new practices related to requirement analysis and negotiation. These include using print-outs of screen shots to clarify conflicts and engaging in other forms of conflict resolution (RA9), and using live case scenarios to support negotiations (RA10).

Table 6.4-2 New Requirements analysis and negotiation Practices - PSIRE

New Requirements Analysis and Negotiation		
No	**RE Practices**	**PSIRE**
RA9	Use print-out of a screen shot to clarify conflicts, and engaging in conflict resolution	Standardised use
RA10	Use live case scenarios to support negotiations	Standardised use

The use of printouts and other forms of conflict resolution and the use of live case scenarios have 'standardised use' in PSIRE. Neither of these two practices has been mentioned in past literature on this topic. Examples of analysts using print-outs of screen shots and using a copy of the software to support their negotiations with a client are given in the descriptive notes below:

While showing the client the software, the analysts also engaged in explaining different elements of the software (***RA10)***. *They explained its various functionalities and clarified various textual and graphical materials. The analysts tried to demonstrate the software in a way that showed how it could support the client's business* ***(RA10)***. *They also trained some users at the client's organisation on using some of the software functions. The analysts showed the client printouts of various software functions, in order to gather further client needs or gain more detail about the client needs they already knew about* ***(RA9)***. *They were therefore using a copy of the packaged software in order to explain the software functionalities, and in order to carry out requirements validation* ***(RA10) [Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]***

The use of print-outs of screen shots assumes importance in PSIRE as this can give users a valuable chance to see how a piece of software looks, and also helps analysts to identify functions that need to change. Meanwhile, previous research by Khan (2011) shows that combining interviews, workshops, and prototyping can be a very effective method for the RE phase.

6.5. Describing Requirements Practices

'Describing requirements' is defined as guidelines to be followed when writing requirements. If good guidelines are established, this will maximise analysts' and developers' understanding of requirements.

Table 6.5-1 Describing requirements in PSIRE

Describing Requirements Practices			
No	**RE Practices**	**Type**	**PSIRE**
DR1	Define standard templates for describing requirements	Basic	Standardised use
DR2	Use language simply and concisely	Basic	Standardised use
DR3	Use diagrams appropriately	Basic	Discretionary use
DR4	Supplement natural language with other descriptions of requirements	Basic	Common use
DR5	Specify requirements quantitatively	Intermediate	Discretionary use

As can be seen from Table 6.5-1, four practices can be considered 'basic' in RE: defining standard templates (DR1), using language simply and concisely (DR2), using diagrams appropriately (DR3), and supplementing language descriptions of requirements with other forms of description (DR4). In some cases they gave me reasons for such use. For example, here is an excerpt from my descriptive analysis, which provides such detail:

The next stage of each Focus Group was to start talking about the 'Describing Requirements' practices that analysts used for packaged software pre-implementation and implementation. I had already defined these practices for the analysts at the start of each Focus Group. We first discussed the practice of defining standard templates for describing requirements (which I called ***DR1****), which is a basic practice in RE. All the analysts who spoke about this stated that this was a practice they always did in their organisation. User5 stated: "We always do that because that's a pretty basic thing. When we describe requirements we all have to use very similar templates so it's not confusing". User4 said "We want to show consistency and also make it possible for other employees to read the document if necessary so we would always use the same kind of template. We figure that out before proceeding ... That is a practice we always use". I next discussed* ***DR2*** *with them, 'use language simply and concisely'. It seemed like there could be some ambiguity about what was considered 'simple' language, but the statements made by the analysts all suggested that they thought this was important and that analysts in their organisations always tried to do this. User2 stated: "We don't want what we're writing to be unclear to anybody and we want to be able to understand our documents later on. So we use technical language but in a simple way, and we keep things short and specific". This statement and others that were very similar strongly suggest that this is a practice that they always use, so I have found it to be a standardised practice* ***[April 26 and May 3 2012 - Organisation 1 & 2 - Describing Requirements Practices].***

Therefore, in PSIRE, most of those practices that Sommerville & Sawyer (1997) and Cox et al. (2009) consider 'basic' RE practices do receive standardised use. However, DR3, using diagrams appropriately, is practiced with discretionary use, probably because the software has already been developed. Another practice can be observed in PSIRE is the use of DR5, specifying requirements quantitatively, which Sommerville & Sawyer (1997) and Cox et al. (2009) regard as 'intermediate'. However, this practice has discretionary use in PSIRE. Information supporting these findings is provided in the excerpt of descriptive analysis below:

We also discussed ***DR3****, "use diagrams appropriately". The diagrams meant here are diagrams that would help to show the connections between different parts of software, what's needed to run the software, or explaining the purpose behind different parts of the software, or functions of different parts of the software. Overall, the analysts said this is not really used that often. User4 said something that seems to explain that this practice isn't used much in packaged software pre-implementation and implementation because the software has already been developed and thus doesn't need a lot more technical explanation at this point, only demonstration: "The analysts are already ready familiar with the software once the package has been developed so we don't really need diagrams that explain it more. We have learnt enough about the software already. Sometimes we make diagrams to show to the users" [User4]. It seems that whether the analysts create or provide diagrams as part of pre-implementation and implementation activities relies on the individual situation and whether the Team Leader of a group of analysts or the group themselves thinks it's necessary. Thus, it seems that this practice in PSIRE has 'discretionary use': it is just used as needed. Then we discussed* ***DR4****, 'supplementing natural language descriptions of requirements with other forms of description'. This means that while the analysts almost always tried to use language simply and concisely they might also have the option to supplement such simple/straightforward descriptions of requirements with other ways of describing the requirements. The* *information provided by the analysts in response to this practice suggested that this practice has 'common use' in PSIRE.* ***DR5****, 'specifying requirements quantitatively' was also discussed. This is a practice that is followed at an intermediate level: that is, it's sometimes used. I had explained that by specifying requirements quantitatively, analysts said that they would only do this if it were really necessary it wasn't a standard practice and it wasn't something that they did very often. Therefore, I believe it has discretionary use in PSIRE* ***[April 26 and May 3 2012 - Organisation 1 & 2 - Describing Requirements Practices]***

Table 6.5-2 New Describing requirements Practices - PSIRE

New Describing Requirements Practices			
No	**RE Practices**	**Type**	**PSIRE**
DR6	Specify relationship between users' needs and other software functions	Basic	Standardised use
DR7	Specify relationship between users' needs and data stores	Basic	Standardised use

Table 6.5-2 shows some new practices that have been identified as being involved in describing requirements in PSIRE. The new practices involve specifying relationships between users' needs and other software functions (DR6), or between users' needs and data stores (DR7). These practices have standardised use in PSIRE. My descriptive notes about these practices are given below:

As in my discussion about the previous set of practices, discussions with the analysts revealed that there were some practices they follow in PSIRE that I had not known about or mentioned on the list. So two new practices were found: ***DR6****, 'specify the relationship between users' needs and other software functions', and* ***DR7****, 'specify the relationship between users' needs and data stores'. We discussed both of these, and the analysts told me that specifying the relationship between users' needs and other software functions meant that the analysts had to have a thorough grasp of the structure of the software and how each function of the software fit together. They had to know why each function was present and whether/how the software would work if that function was taken away. Various analysts said this was really important because sometimes clients state that one of their requirements is to get rid of a specific function: "they don't think they'll need or use that function and they don't want it there" [User4]. The analysts have to consider and be able to figure out whether they can follow the client's request to remove a particular function, because if "we just go ahead and remove it, this could have repercussions for the software, making it run worse, removing information or fields that could be needed elsewhere in the system, or making the software inoperable" [User5].* ***DR7****, specifying the relationship between users' needs and data stores, relates specifically to thinking about the problems that could occur if particular fields and functions are added or removed, in relation to whether this would add/remove data that is needed elsewhere in the system or that would need to be accessed by the client from another part of the system/another screen. Sometimes the analysts will realise that the request will not disrupt the software, so they may go ahead with it; other times they decide it is impossible or too risky* ***[April 26 and May 3 2012 - Organisation 1 & 2 - Describing Requirements Practices]***

The importance of such issues related to specifying the relationship between users' needs and other software functions and specifying the relationship between users' needs and data stores is also shown in my analysis and comments about the 'Client Request Form/Misalignments Specification Form' provided below. In the case that a client wishes to make a request for a particular function to be added or for another to be removed, they fill out a particular form and send it to the software development organisation. The form will state what exactly is requested and why the request has been made. The analysts use such forms in order to keep track of how the client's requirements are related to business needs and to aid them in considering whether a request is possible. See the discussion of the 'Misalignments Specification Form' below:

> ***Transaction Formula:*** *80 JD is taken from the first month of an employees' salary; return the 80 JD after 3 months if the employee is still active or minimum working days is 70* ***(DR6)****. The deduction of costs for uniforms should be saved in deduction table under uniforms deduction* ***(DR7)***
> ***Relation with other functions:*** *Employees Salary, employees' status, salary report* ***(DR6)****.*
> ***Database Tables:*** *add uniforms deduction field to deduction table* ***(DR7)***
> *In this case, the request is created as the analyst fills in a request form by using a software tool that was created internally by the analysts' company. This software tool is used by the analysts to exchange information within the analysts' company and to manage their work. Even without having any prior knowledge of the analysts' organisation involved, one can gain insights about the analysts' organisation by making observations of the request form. The context and layout of the request form is very structured and the language used on the form is formal* ***[Feb 14, 2012/ Organisation 2 - Misalignments Specification Form]***

In PSIRE, specifying the relationship between users' needs and other software functions (DR6) involves task activities such as discovering what functions are redundant in software and what software functions are essential. This is important in PSIRE because when users inform the analyst that a particular function is redundant and that they would like it removed, the analyst has to consider whether the unwanted function is actually connected to other functions of the software. This process also involves identifying which customisation requests can be met without disrupting the software. Thus, users' needs, the scope of the project, and customisation risk are all factors that are considered when dealing with the intersections between software functions and users' needs.

6.6. System Modelling Practices

System modelling is a process activity that relates to the building of abstract system models that aid in the understanding and analysis of requirements and of understanding their implications for the proposed system. System modelling may also follow various guidelines and can be carried out at basic, intermediate, or advanced levels.

Table 6.6-1 System modelling in PSIRE

System Modelling Practices			
No	**RE Practices**	**Type**	**PSIRE**
SM1	Develop complementary system models	Basic	Never used
SM2	Model the system's environment	Basic	Never used
SM3	Model the system architecture	Basic	Never used
SM4	Use structured methods for system modelling	Intermediate	Discretionary use
SM5	Use a data dictionary	Intermediate	Common use
SM6	Document the links between stakeholder requirements and system models	Intermediate	Standardised use

From Table 6.6-1 we can observe that developing complementary system models (SM1), modelling the system's environment (SM2), and modelling the system's architecture (SM3) are practices used at the basic level in RE, but that they are never used within PSIRE. It is also interesting to note that a basic system model in PSIRE would be similar to a prototype method with natural language description (Beecham et al., 2003). However, formal system modelling such as context diagram, DFD, use cases, and so on have not been used as requirements modelling methods in PSIRE (Laplante et al., 2002). This may be because PS implementation analysts focus on customisation requests and their impact on software functions. Rather than collect all customisation requests at one time and work on them all at the same time, analysts using PSIRE collect new requests as they come in, possibly on a daily basis. As a result of this approach, they do not engage in system modelling.

Other forms of system modelling practices are practiced at varying levels within PSIRE. Documenting the links between stakeholder requirements and system models (SM6) is a practice that has standardised use in PSIRE, but using a data dictionary (SM5) and using structured methods for system modelling (SM4) are not standardised practices in PSIRE. Rather, the data dictionary receives common use (SM5) and the structured methods (SM4) receive discretionary use. However, the way that analysts document the links between stakeholder requirements and system models (SM6) in PSIRE is by using a Users' needs/Misalignments Specification document in which the relationships between users' needs and other software functions are specified. The document also specifies the relationship between users' needs and data stores.

6.7. Requirements Validation Practices

Requirements validation can be defined as consisting of practices that make up formal validation procedures that help analysts to check for problems related to incomplete requirements, inconsistent requirements, or incompatibility between systems or between an organisation and a new system. Requirements validation practices are also established to ensure that requirements can be verifiable and to help set quality standards.

Table 6.7-1 Requirements validation in PSIRE

Requirements Validation			
No	**RE Practices**	**Type**	**PSIRE**
RV1	Check that the requirements document meets your standards	Basic	Standardised use
RV2	Organise formal requirements inspections	Basic	Discretionary use
RV3	Use multi-disciplinary teams to review requirements	Basic	Discretionary use
RV4	Define validation checklists	Basic	Common use
RV5	Use prototyping to animate requirements	Intermediate	Standardised use
RV6	Use a draft user manual	Intermediate	Standardised use
RV7	Propose requirements test cases	Intermediate	Standardised use
RV8	Paraphrase system models	Advanced	Discretionary use

As seen in Table 6.7-1, in RE, there are four 'basic' practices involved with requirements validation, which are checking the standard of the requirements document (RV1), organising formal requirements inspections (RV2), using multi-disciplinary teams when reviewing requirements (RV3), and defining validation checklists (RV4). Within PSIRE, however, these practices receive varying levels of practice. These findings regarding requirements validation practices were drawn primarily from my day by day descriptive analysis notes. For example, here is an excerpt from my descriptive notes:

*I first discussed the practice of checking the standard of the requirements document **(RV1)** with the analysts. The different responses I received suggested that this is a practice they always observe; they see* *it as a basic expectation. The analysts also gave me some ideas regarding the kinds of things they are looking for when they check the requirements document to see if it meets the necessary standard: User2 "We are looking for incomplete requirements, for places where the requirements don't make sense, for places where we might have repeated the requirements". It seemed that the analysts did not very often organise formal requirements inspections* ***(RV2)****: they said that this is seldom done and only organised if the Team Leader thinks it is needed. One analyst told me "we don't usually need formal inspections because we already have the requirements document and sales and marketing team report which tells us what to provide"* ***(RV3)****, using a multi-disciplinary team to review requirements, also came through as a practice that is not used very much. The analysts do not think this is very necessary because analysts' concentre on how the requirements and users' needs related and impact on packaged software functions only so they used Misalignments specification form to express users' needs. The analysts did say that they often defined validation checklists* ***(RV4)****. Different analysts during the Focus Groups considered defining the* ***validation checklists*** *important "once we collected many requirements from users at one meeting so we have to list these requirements down to be clear about users' needs [User5]* ***[May 12 2012 - Organisation 1 & 2 - Requirements Validation]***

While organising formal requirements inspections (RV2) and using multi-disciplinary teams when reviewing requirements (RV3) are both basic practices in RE, they are practices that have only discretionary use in PSIRE. This is because with packaged software there can be different sources of requirements for a computer-based system: for example, end-users of the system, managers in the organisation and customers of the organisation. All have their own viewpoint on the services that the system should provide. During a requirements review process these different viewpoints should be considered in order to reduce requirements errors. It may be that RV2 and RV3 should be standardised practices for large organisations where many systems are complex and difficult to quickly understand. In this study, however, RV2 and RV3 are not standardised practices, because most of the client organisations considered are SMEs in which the number of users is small. The analysts engaged in PSIRE instead used software demonstrations to target users and to collect the misalignments at the same time as validating the misalignments. Thus, answering the customisation requests became part of their daily work and projects. Meanwhile, defining validation checklists (RV4) is a practice that may be widely used in PSIRE, but is not mandatory. Validation checklists concentrate on the requirements document as a whole and help those conducting validation to concentrate on important attributes of the requirements document. However, defining validation checklists was not considered a standard or essential practice by organisations engaging in PSIRE.

The following excerpts taken from my descriptive analysis of two of the software demonstrations I attended show analysts engaging in requirements validation practices RV1 through RV4 so that they can collect information about misalignments, validate the misalignments, and convince their clients about the suitability of the software at the same time:

As the analysts showed the client the software and discussed the client's needs from the software, it was apparent that the analysts were trying to validate the client's needs ***(RV1)*** *(i.e. confirm those needs/requirements they had already collected), to get a better understanding of the client's business process, and to identify any mismatches between the client's needs and the software. These actions helped the analysts know more about the client's requirements* ***(RV2)****. While showing the client the software, the analysts also engaged in explaining different elements of the software* ***(RV3)****. They explained its various functionalities and clarified various textual and graphical materials [****Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs****]*
…….The analyst then carried out a software demonstration on his own to validate that the modifications had been made and to show how the software functions worked ***(RV1)****. Then, as part of a live scenario demonstration, he got the client to use the software as though they were issuing books, and trained them on how to use the software functions* ***(RV4)****. The client was happy with everything that had been done and with the training they had received, so at the end of the meeting the client signed the modifications acceptance form and the trained functions list..... [****March 21, 2012 - Organisation 1 - During Implementation – Software demonstration & validation of users' needs****]*

Three practices in RE validation are carried out at the intermediate level: using prototyping to animate requirements (RV5), using a draft user manual (RV6), and proposing requirements test cases (RV7). Within PSIRE, these three practices all receive standardised use. Within PSIRE, using prototyping to animate requirements (RV5) and proposing requirements test cases (RV7) involve software demonstration. One last practice, paraphrasing system models (RV8) is treated as an advanced practice in RE and has only discretionary use in PSIRE. The excerpt from my descriptive notes below shows several of these requirements validation practices being used in PSIRE:

We discussed using prototyping to animate requirements. I discovered that this is something the analysts do very often; in fact, they said it was a standard practice for them ***(RV5)****. The analysts from both organisations always create prototype versions of the software (which are completed packages but possibly without all of the functions the client might require) so that they can do a live demonstration of the software to the client. They saw this as the best way to present information about the software to their clients, since clients can see it working* ***(RV5)****. Proposing requirements test cases was also a practice that different analysts said they followed every time. One analyst said that this was always done because "users would like to see how their needs and requests are working with really data so we present the modifications by really data from users business process"[User2].I also learnt from discussions during the Focus Groups that analysts from both of the organisations always use a draft user manual while conducting requirements validation. The draft version of a user manual is taken along to any software demonstration that analysts give because it can be used to help explain the software to clients or to do some initial training on the software. User4 said that "we leave a copy of the draft manual with the client* *to look at it so that clients will get information about software functions and how they work and we usually give our clients copies of draft user manuals after clients accept the software offer"* ***(RV6)****. The draft manual is given to the client to read or keep at some point. I also asked the analysts if they paraphrased system models, which is another practice sometimes used in RE. The analysts' statements suggested that this practice only receives discretionary use. One analyst from Organisation 2 told me "We don't usually do that unless users' needs are added some benefits for our software otherwise the we just add as modifications on the software within client profile" [User5]* ***[May 12 2012 - Organisation 1 & 2 - Requirements Validation]***

I observed analysts from both organisations carry out these practices (RV5 through RV8), as they prepared for and carried out software demonstrations:

....The analysts tried to demonstrate the software in a way that showed how it could support the client's business ***(RV7)****. They also trained some users at the client's organization on using some of the software functions. The analysts showed the client printouts of various software functions* ***(RV6)****, in order to gather further client needs or gain more detail about the client needs they already knew about. They were therefore using a copy of the packaged software in order to explain the software functionalities* ***(RV7)****, and in order to carry out requirements validation* ***(RV5)****.....* ***[Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]***
....Then they started explaining what modifications they had made to the software functions ***(RV5)****. Their next task was to demonstrate the software to actually show that these modifications had been made successfully and that they reflected the client's requests* ***(RV5 and RV7)****, and to show the software actually working* ***(RV7)****....* ***[March 12, 2012 – Organisation 1 - During Implementation – Software demonstration & identify users' needs]***

The way that the analysts showed their client the software in order to demonstrate that completed modifications worked successfully matches recommendations in the literature. It was recommended by Beecham et al. (2003) that for each requirement one or more test cases should be proposed to identify any requirements errors. Requirements errors can have a wide impact on the success of software development projects. No software process can keep delivery times, costs and product quality under control if requirements errors are not identified and removed early on (Beecham et al., 2003). One solution recommended for managing uncertainty about requirements is prototyping. Other motivations for building a prototype are: eliciting requirements, validating requirements, and determining the feasibility of particular solutions (El Emam & Madhavji, 1995; Beecham et al., 2003). However, in PSIRE, using prototyping and requirements test cases through software demonstration were also used as strategies for identifying 'actual' misalignments and 'perceived' misalignments. This is understandable because analysts support the idea of minimising customisation. My descriptive analysis below discussing their approach toward customisations demonstrates this:

> *... [The] Team Leader said, analysts believe that a misalignment is 'real' only when the software provided does not support the transaction that has been requested or does not support necessary transaction formula. If the software functions actually do support some transaction desired by the client, but do the transaction in a different order than what the client wants, the analysts do not consider this a real misalignment: instead they consider it only a 'perceived' misalignment – i.e. a misalignment from the point of view of the client (this done by software demonstration **RV5 and RV7**). The analysts seemed to be less interested in correcting misalignments that were only 'perceived' ones rather than 'actual' ones **(RV5)**…….From my general interaction with this Organisation (and also with Organisation 1) I can see that these two medium-sized software companies don't really want to carry out many customisations. Both organisations have a policy of minimising customisation as much as possible. This explains why the analysts from Organisation 2 were quick to show their client that their software actually did provide one of the transactions that they wanted **(RV5 and RV7)**. I've concluded that these medium-sized software companies want to minimise customisation as much as possible, and that when they consider whether to go ahead with a customisation, they are less likely to do it if a) the customisation request is outside of the software scope, and b) the misalignment that the client identified is only a 'perceived' misalignment. I think that these guidelines for making decisions about misalignments might sometimes be affected by how much the client is willing to pay for a customisation…**[Feb 27, 2012 - Organisation 2 - Respond to the discovery of a misalignments]***

6.8. Requirements Management Practices

Requirements management is defined as a system of guidelines and activities used to manage requirements information throughout the project development life-cycle.

Table 6.8-1 Requirements management in PSIRE

Requirements Management			
No	**RE Practices**	**Type**	**PSIRE**
RM1	Uniquely identify each requirement	Basic	Common use
RM2	Define policies for requirements management inspections	Basic	Standardised use
RM3	Define traceability policies	Basic	Discretionary use
RM4	Maintain a traceability manual	Basic	Discretionary use
RM5	Use a database to manage requirements	Intermediate	Standardised use
RM6	Define change management policies	Intermediate	Standardised use
RM7	Identify global system requirements	Intermediate	Discretionary use
RM8	Identify volatile requirements	Advanced	Discretionary use
RM9	Record rejected requirements	Advanced	Standardised use

It can be seen from Table 6.8-1 that requirements management involves four basic practices: uniquely identifying each requirement (RM1), defining policies for requirements management inspections (RM2), defining traceability policies (RM3), and maintaining a traceability manual (RM4). However, three of these practices (RM1, RM3, and RM4) are not practiced at a standardised level in PSIRE. Instead, in PSIRE, identifying the requirements and defining the policies for requirements management inspections have common use, while defining the traceability policies and maintaining a traceability manual are practices that are not standard or normal, but rather, discretionary. Some observations that helped lead to these findings are shown in the following excerpt from my descriptive notes:

I discussed the practice of uniquely identifying each requirement with the analysts. It turned out that even though this is a basic practice in RE, it isn't a practice that the PSIRE analysts use every time. When I asked them why this is not a standardised practice, I received answers such as: "Doing that practice isn't necessary every time because sometimes the requirements for a piece of software are quite simple or are close to the pre-completed software already" ***(RM1)*** *(User2), and "Some requirements can be grouped together, so we don't have to uniquely identify each requirement"* ***(RM1)****. Next we discussed defining the policies for requirements management inspections, and, from the information the analysts gave, it seems like defining these policies is also done most of the time and could be considered as having a standardised use. When the analysts 'define the policies' they usually consider things like software scope, software prices, and assessment criteria for software implementation* ***(RM2)****. We also discussed two practices that I had listed related to maintaining traceability of requirements. I asked the analysts about 'defining traceability policies' and 'maintaining traceability manual', and found out that the analysts don't always do either of these. Neither of them are done often at all; they are just used when required. User5 said that "we take a more ad hoc approach that instead involves continuous improvement of our product in response to clients' requirements and how clients feel about the product"* ***(RM3, and RM4) [May 17 2012 - Organisation 1 & 2 - Requirements Management]***

As shown below, in an example taken from my daily descriptive analysis, the analysts I observed often did uniquely identify each requirement (RM1 and RM2). In the example below, we can also observe the analyst using a modification request document in a way that assists with change management (RM6):

The client decided to make a modification request regarding the attribute needed to collect family medical history, so modification request documents were filled out ***(RM1 and RM2)****. The client then signed these documents. The analysts told me that these documents are used in order to formally keep track of what the client had requested – the documents work as validation of the user's needs and also provide some security against the client later asking for further changes* ***(RM6)****. Before the modification request documents were signed by the client, there was a short discussion about what the likely time estimate of the modification would be* ***[March 12, 2012 – Organisation 1 - During Implementation – Software demonstration & identify users' needs].***

One of the practices that are intermediate in RE, using a database to manage requirements (RM5) is a standardised practice in PSIRE. For example, the organisation in the quote below made sure to keep a database in which they documented installations of software and any changes made to software:

In this case, the request is created as the analyst fills in a request form by using a software tool that was created internally by the analysts' company ***(RM5)****. This software tool is used by the analysts to exchange information within the analysts' company and to manage their work ...* ***[Feb 27, 2012 - Organisation 2/ Respond to the discovery of a misalignment]***
At the end of the meeting, the clients signed a report stating that the installation had taken place. It appears that the analysts make sure to document every action of this kind as a way of managing the progress of different projects ***(RM5) [March 11, 2012 - Organisation 2 – Installation]***

However, one practice that is intermediate in RE, identifying global system requirements (RM7), receives only discretionary use in the PSIRE process. During the Focus Groups I held, I was told by analysts that 'identifying global system requirements' was not a high-priority practice because the analysts rarely had to do this. They were not usually considering global system requirements since they deal with local users. Lastly, two practices can be considered advanced in RE: identifying volatile requirements (RM8) and recording rejected requirements (RM9). In PSIRE, the practice of identifying volatile requirements is completely discretionary, while the practice of recording rejected requirements is not an advanced practice, but rather a standard one that is always used. The finding regarding the facts that analysts did always keep records of which requirements had been rejected was drawn primarily from information located in my day by day descriptive analysis forms. Here is a relevant excerpt from my descriptive notes:

The analysts ended up signing off on the 'change request' that the clients had made regarding the formula related to uniforms ***(RM6)****. To me, it looked like there were two main benefits to having such a 'change request' form and needing to have it signed: it would be useful to have an official documented request to refer back to later, for information or for confirmation that an agreement was reached, and this request form helped to manage the whole implementation process. The analysts did turn down one customization change request, though. The Accounting Manager of the client organisation asked the analysts to add some attributes to the employee' salary reports. The analysts answered this by saying that the attributes that were being requested already existed within the software* ***(RM9)****. At the end of the meeting, the analysts signed off the users' meeting summary* ***[Feb 26, 2012 – Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]***

In RE, analysts may also record rejected requirements (RM9). If this is done, it is practiced at the advanced level. However, in PSIRE, recording rejected requirements is actually treated as a core practice and has standardised use. Analysts carrying out PSIRE may have a greater need than analysts doing RE to record rejected requirements. This is because analysts doing PSIRE may need to refer back to a list of requirements that other analysts have rejected in response to requests from other clients. This can tell the analyst which requirements were previously rejected because they would have a negative effect on the software. Alternatively, analysts may look at such a list in terms of considering possible benefits that could be gained if the rejected requirements were followed up on in future. For example, the rejected requirements could be kept in mind for inclusion in the next release requirements. There are some factors that may lead analysts to reject requirements, such as whether a requirement is 'actual' or 'perceived', the potential benefits of the requirement, the software scope, software price, and the client's organisation size.

6.9. Result Validation through Participants Checking

This section explains and validates the study findings from the participants' perspective. It is essentially an assessment of the RE practices in terms of PS implementation. After the data had been analysed inductively to produce the findings reported in this and the preceding chapter, I sent emails to both organisations in order to arrange participants checking sessions with analysts. One organisation agreed to participate in participants checking. In 'stage one', the participants, consisting of 16 analysts, were asked to review a checklist (Appendix H) of RE practices and mark those practices which they either personally used or witnessed someone else on their team using in the process of PSI, and to describe the way it was used (either 'SU', 'CU', 'DU', or 'NU').

The instructions for the participants stated: "We are interested in understanding which practices you use in requirements engineering. For each practice shown in the following tables place a tick in the column that indicates whether you either personally used or witnessed someone else on your team use that practice during the packaged software implementation process".

- Standardised use (SU): This practice has a documented standard and is always followed as part of the organisation's software development process i.e. it is mandatory.
- Common use (CU): This practice is widely followed in the organisation but is not mandatory.

- Discretionary use (DU): This practice is used at the discretion of individual project managers. Some may have introduced the practice for a particular project.
- Never used (NU): The practice is never or rarely applied.

In ‘stage two’, I compared the practices marked in order to note similarities and differences between my ethnographic account analysis and the participants’ answers, as shown in Table 6.9-1 and Table 6.9-2 below. I adopted an acceptable matching level and compared what the organisation’s analysts said with what I had originally found, in order to display whether the two forms of results matched or not. Table 6.9-1 and Table 6.9-2 both show requirements documentation practices, and function as examples of my result validation of requirements engineering practices. Further comparisons showing result validation can be found in Appendix H.

Table 6.9-1 Requirements documentation Practices – Result validation

Requirements Documentation Practices				
No	**RE Practices**	**Type**	**PSIRE**	**Validation**
RD1	Define a standard document structure	Basic	SU	SU
RD2	Explain how to use the document	Basic	CU	SU - CU
RD3	Include a summary of the requirements	Basic	SU	CU - SU
RD4	Make a business case for the system	Basic	SU	SU - CU
RD5	Define specialised terms	Basic	DU	SU
RD6	Make document layout readable	Basic	CU	CU - SU
RD7	Help readers find information	Basic	CU	SU - CU
RD8	Make the document easy to change	Basic	CU	CU

As shown in Table 6.9-1, during my ethnographic research I made observations of a range of basic RE practices carried out as part of PSIRE. Some of these results were validated during the original focus group sessions with analysts. For example, RD1, ‘defining a standard document structure’ was found to have standardised use (SU) during my ethnographic analysis and was confirmed by analysts as having standardised use in their company. RD8, ‘making the document easy to change’, had the same result, appearing as CU in my ethnographic results and as CU from the perspective of the analysts.

As can be seen, there were some differences, however. During ethnographic research, I found that RD5, ‘defining specialist terms’, was practiced with discretionary use (DU). However, analysts marked it as receiving standardised use (SU). The remaining five practices all received ratings from the analysts that partly matched and partly diverged from my original findings. For example, I had found RD4, ‘making a business case for the system’, to be practiced with SU, whereas analysts suggested it was practiced with both SU and CU (with SU being the slightly more common response on the checklist). A similar result happened with RD6, ‘making the document layout readable’: I had observed this to be CU, but while the majority of analysts did mark this practice as CU, a similar number also marked it as SU.

A similar pattern emerged with the results for RD2, RD3, and RD7, only with these the diverging result turned the importance of two levels of practice the other way around. For example, my ethnographic observation of how RD2 (‘explain how to use the document’) was practiced, found that it was practiced with CU. During validation by analysts, some analysts did mark it as receiving CU, but a slightly higher number marked it as SU, thus returning a result of SU-CU. RD7 had the same result, and RD3 the same result - however, in reverse.

Table 6.9-2 New Requirements documentation Practices – Result validation

New Requirements Documentation Practices PSIRE				
No	**RE Practices**	**Type**	**PSIRE**	**Validation**
RD9	Users’ needs/Misalignments specification document	Basic	SU	SU
RD10	Estimating time needed for users’ needs document	Basic	SU	SU
RD11	Estimating cost needed for users’ needs document	Basic	SU	SU - CU
RD12	Users’ needs validation document	Basic	SU	SU - CU

Table 6.9-2 shows the same form of comparison carried out in relation to new requirements documentation practices. In this case, the responses given by analysts when marking use of the requirements RD9 (‘creating a users’ needs/ misalignments specification document’) and RD10 (‘estimating the time needed for creating a users’ needs document’) validated the results gained from ethnographic analysis; for each, the

PSIRE result was SU, and the validation result was also SU. However, the validation result for RD11 ('estimating the cost associated with creating a users' needs document') was different than the result found during the ethnographic observations. The PSIRE result was SU, but while the slight majority of analysts marked it as SU, a similar number marked it as CU, thus providing the result of SU-CU. The same difference was observed between the PSIRE and validation results obtained for RD12, the practice of 'including the users' needs validation document'.

The analysts' responses to questions about requirements elicitation practices showed that in most instances, they agreed with my findings of how each requirements elicitation practice was used. For example, in their ratings of how 13 different requirements elicitation practices were used, their answers differed strongly from my ethnographic observations in only 2 instances. While I found that 'collecting requirements from multiple viewpoints' (RE9) received DU, the analysts viewed it as having SU. While I found that 'defining operational processes' (RE12) also had DU, the analysts marked it as a practice that had CU. Analysts also returned answers that diverged slightly from mine with regard to five other requirements elicitation practices: as happened with how analysts responded to questions about requirements documentation practices, in five cases the analysts' marked the checklist in such a way to return a result only slightly in favour of one form of use. For example, while I had found 'reusing requirements' (RE13) to have SU in PSIRE, a majority of analysts marked it as having SU but almost as many marked it as having CU. The validation result returned was therefore SU-CU. Four other responses were split in a similar way, returning results that partly corresponded with my ethnographic findings.

Other results followed a similar pattern: out of 8 questions about Requirements Analysis and Negotiation Practices, 6 answers returned completely agreed or partly agreed with my findings. Only 2 gave a very different result. In the case of System Modelling Practices, in response to 6 questions, only 1 answer by analysts returned a completely different result from my ethnographic findings, and in the case of Requirements

Management Practices, only 1 of 9 responses made by the analysts did not agree in any way with my own findings.

The general result of the validation process carried out by means of the checklist was that the viewpoints the analysts held of the practices involved in PSIRE had an approximately 86% similarity with my own view of the process. Therefore, the findings made through my analysis of my ethnographic account were validated by the analysts at a rate of about 80-90%.

6.10. Summary

In this chapter, I have adapted the lists of RE practices and frameworks used by Sommerville & Sawyer (1997) and Cox (2009) in earlier studies, making use of these frameworks to assess packaged software implementation RE (PSIRE) practices. RE practices were assessed in terms of whether they were used at all in PSIRE, and, if used, to what degree they were used.

Splitting the whole process of implementation into specific elements and activities, according to the use of particular documents or particular groupings of processes for collecting and validating requirements, I then analysed the level of use of each practice in PSIRE by making use of the four levels of assessment developed by Sommerville & Sawyer (1997): standardised use, common use, discretionary use, and never used. The assessment sections of RE considered were: requirements documentation practices, requirements elicitation practices, requirements analysis and negotiation practices, describing requirements practices, system modelling practices, requirements validation practices, and requirements management practices. After an initial analysis of the results gained from this assessment, the results were validated by means of a checklist provided to many of the same analysts who had been observed and interviewed during the ethnographic fieldwork I completed in Jordan.

In general terms, it was found that PSIRE introduced new methods of documentation, was not as concerned as general RE practice with looking for domain constraints or with collecting requirements and viewpoints from multiple sources, was more likely to involve live software demonstrations and screenshots to validate user needs, and was more likely to involve the compilation of a user manual. In PSIRE, prioritising requirements is not a basic practice; instead, analysts collect requirements in a circular process, with managers then directing analysts regarding which requirements to direct most attention toward. PSIRE was also found to place emphasis on assessing requirements risks and on considering the relationship between users' needs and the inter-relationships between software functions, as analysts engaging in PSIRE do not wish to disrupt functions of their software when making modifications in response to client requests.

The assessment of PSIRE led to the discovery of new RE practices that were being used in four of the areas examined. New RE practices were found in relation to requirements documentation, requirements elicitation, requirements analysis and negotiation, and describing requirements. Explanations were also provided of how and why these new practices were introduced and used. While the investigation of system modelling, requirements validation, and requirements management did not reveal new RE practices in PSIRE, the discussion provided in this chapter explains why certain practices receive an emphasis in PSIRE.

The results of my assessment of practices used in PSIRE were then validated by means of participants checking held with analysts from one of the organisations involved in my ethnographic study. By means of a checklist completed by analysts, I was able to compare my observations about PSIRE and my analysis with the viewpoint of these packaged software analysts. The results of the validation process showed that the analysts participating in the validation agreed with my analysis of how they practiced PSIRE, at a rate of approximately 86%.

In this chapter, I provide the first detailed explanation in RE literature of why some new requirements engineering practices are introduced in PSIRE, and also provide valuable explanation of why certain other RE practices also used in PSIRE acquire an increased emphasis in PSIRE.

Chapter 7 Discussion

7.1. Introduction

This study was motivated to fill a void in the literature regarding the understanding of packaged software implementation requirements engineering. Just as the general topic of packaged software suffers relative neglect in the IS literature, so the understanding of the requirements engineering process to implement packaged software demands greater attention. Research studies in related areas such as RE practices, critical success factors in packaged software implementation, and misalignments that appear during implementation provide little guidance to researchers interested in packaged software in terms of requirements engineering at small packaged software vendors (SPSVs). Given these shortcomings, I conducted an ethnographic study in two software development companies who create and implement packaged software.

This approach to explaining packaged software implementation from the perspective of the SPSVs provides a different understanding than is provided by many previous studies. These studies have typically focused on the users' perspective of packaged software implementation; a perspective that revolves around their attempt to select packaged software that fits their process. By changing the perspective used from 'outside' the SPSVs to 'inside' the SPSVs, this research provides new insights. It also highlights the need for an extension of the current research agenda. If researchers begin to shift their focus from examining users' organisations to examining SPSVs' views of packaged software implementation, we will gain a more complete understanding of all of the sites and participants involved with packaged software implementation.

Previous research on RE practices at SPSVs has tended to investigate the provision of bespoke software, and has been dominated by software development studies (El Emam & Madhavji, 1995; Nikula et al., 2000; Aranda et al., 2007; Quispe et al., 2010; Merten et al. 2011). In addition, there have been some studies related to the development of packaged solutions, such as those reported by Daneva (2004), Barney et al. (2006), Daneva & Wieringa (2006), Lehtola & Kauppinen (2006), Daneva (2007), and Karlsson et al. (2007).

In contrast, this study has addressed how SPSVs carry out packaged software implementation, specifically focusing on requirements engineering practices in that context. The Findings and Discussion highlight some of the dynamics and complexity that these SPSVs face, and their reactions to the associated challenges that arise. Putting the organisations and organisational practices at the centre of attention, this research advances our understanding of packaged software implementation from the point of view of the SPSVs as provider/producer, and its views about requirements engineering practices

7.2. Research Questions Revisited

Packaged software implementation (PSI) is an IS topic that has been largely disregarded in academic literature. In particular, studies of PSI from a requirements engineering practices perspective and from the perspective of (the analysts at) a packaged software development company have not yet been carried out. Investigating packaged software implementation from a requirements engineering practices perspective is not a novelty; however this thesis provides the first known ethnographic study of packaged software implementation, investigating in depth the day-to-day practices of analysts.

The following research questions emerged as a result of identifying gaps in the literature and after having reviewed data collected in the field (see Chapters 1 and 2). These research questions highlight the challenging relationship between users' needs and packaged software functionalities, and relate to the issue of how misalignments appearing during PS implementation may be dealt with, and how participants in PS implementation can achieve a better fit through RE practices.

In the context of packaged software implementation by small packaged software vendors in Jordan:

- What are the analysts' practices?

- How do analysts conduct these practices?
- Why do analysts conduct these practices?

In the following sections, I provide possible answers to these research questions as have emerged from the data and analysis presented in Chapters 5 and 6. My discussions of these research questions highlight some of my main findings in the context of PSIRE, particularly relating to the: (1) Packaged software pre-implementation; and (2) Packaged software during implementation.

7.3. Packaged Software Pre-implementation

The pre-implementation stage investigated in this study resembles such feasibility studies as those used in traditional RE, at a high, abstract level. This is because feasibility studies in traditional RE and the pre-implementation stage discussed here are similar in terms of their purpose, such as determining software objectives, time and budget. However, at the practical level, pre-implementation practice has its own specific form and nature.

The first perspective from which the pre-implementation stage may be considered is that software packages may serve as surrogates for existing systems:

…….They then used a real case scenario of trying to place an items order so they could show how the software would let the client 'add item' and make an item order. For example, they showed the client the interface for making items orders, and they showed the client what would happen. The client actually ***wanted to have a choice where particular items orders were sent to and therefore needed to have reports in the system.*** *During the live scenario* ***the analysts showed that the software had this capability****. The whole time that the analysts* ***demonstrated the software they kept on linking the product's functions to the client's business process by connecting some of the client's concerns about business process (i.e. things that the client said had gone wrong before and various things that they wanted the software to do)*** *to what the software could do. They also discussed the client's inventory business process with the client. The end result was that they managed t****o show the client that the product functions could cover the client's business.*** *There were people present on the analysts' team and. The people sent by the client were the manager of the inventory. During the whole time, the client seemed pretty happy or impressed by the software and at the end of the demonstration the analysts suggested that they thought the client would probably buy it* ***[Feb 1, 2012 - Organisation 1- Pre – Implementation/ Software demonstration]***

A second perspective suggests that a software package may form the whole or part of the recommended solution. However, analysts should consider which package they put on offer very carefully. By offering the most suitable package, they are likely to boost a client's confidence in the recommended package and help to limit the number of changes to the package that the client may request or try to carry out after purchase:

> *…….Before we went to the software demonstration, the analysts told me that the purpose of their software demonstration was to clarify the expectations that the client had already mentioned they wanted from the* ***software and better discover what the client really wanted****. Therefore the meeting and demonstration would be used to understand the project scope. The analysts wanted to know this* ***early on as it would help them to know what the client really wants.*** *They thought that clarifying the whole project scope first would* ***help them discuss specific issues with the client and then be more certain when doing the work.*** *They felt that defining the project's scope* ***would depend on their finding out about the client's problems, needs, and deliverables****..... The comments and questions that the client directed toward the analysts were sometimes related to* ***the product functionalities that the analysts were presenting, but more often were related to the problems that the client was actually facing at the moment*** *– and this led to quite a lot of in-depth discussion of those problems* ***[Feb 8, 2012 - Organisation 2 - Pre – Implementation/ Software demonstration]***

A third perspective about pre-implementation suggests that convincing the client to buy into the PS may possibly lead to reduced resistance to the PS implementation:

> *....it was also a good way of clarifying what the client really needed and gaining more information that* ***would help them deliver a suitable product, since they would see the client's work environment;*** *since using a real case scenario simulates a real situation,* ***it is effective in convincing a client that a product's functions are suitable*** *– i.e. the analysts would be able to show the software actually dealing with requests that the client would typically give it in real life* ***[March 7, 2012 - Organisation 2- Pre – Implementation/ Software demonstration]***

This PhD, therefore, explains and represents how two SPSVs conduct a feasibility study. It highlights in particular: (1) analysts' roles during pre-implementation; (2) software demonstration utilising a live scenario; and (3) mechanisms of scoping and creating a packaged software offer.

7.3.1. Analysts' Roles during Pre-Implementation

The main new finding of this study in terms of the role played by analysts is that during pre-implementation, it is the analysts, as opposed to the sales team, who usually carry out the task of conducting a software demonstration for the client. The software companies observed during the ethnographic study both preferred to have the analysts demonstrate the software. This appears to have been deemed most suitable due to a belief that it is the analysts who know best how the software works and how it has been built, and because analysts are more able than members of a sales team to explain to a client how the software solution offered can solve their problems. In fact, in one of the companies observed, the sales team had previously carried out a software demonstration and had provided wrong information about the software to the client. The companies observed also considered that analysts should carry out the software demonstrations because analysts are more capable than sales team members at responding to requests for new requirements or to any changes to requirements made by the client during the software demonstration. Because of such factors, having analysts carry out software demonstrations had been established as a company strategy.

It should be kept in mind that Jantunen (2010) also found that there is a risk related to having sales team members trying to sell features or accepting customer requests to add new features to packaged software when the features have not actually been developed yet. Such actions basically force the software company to include those features (Jantunen, 2010).

One main new finding, therefore, regarding the role of the analyst in pre-implementation is that the analyst has greater involvement in demonstration of the solution, being expected to conduct such software demonstrations. The analyst involved in pre-implementation is also expected to have some understanding of business concerns and of how to engage in marketing. The analyst doing pre-implementation has more of a hybrid 'analyst-sales-marketing' role and must have the soft skills needed for a software demonstration, which requires presentation skills, communication skills, and sales skills. The analyst is no longer only concerned with software analysis but also with the business dimension of creating software (Jantunen, 2010; Jebreen et al., 2013a; Jebreen et al., 2013b).

7.3.2. Software Demonstration Utilising a Live Scenario

Packaged software development companies have a choice of how to respond to a software demonstration request. Analysts at a packaged software development company may be able to offer more than one solution to the client. In such a case, analysts then need to choose which solution is the preferred one to offer to the client (El Emam & Madhavji, 1995; Al-Mashari, 2003; Haines, 2009). In order to make such a decision, analysts hold meetings that involve themselves and the sales and marketing teams. The limitations of the work domain of the analysts' company are considered (Jantunen, 2010; Jebreen et al., 2013a). Other factors that are considered relevant to making a decision about the solution to offer include the size of the client's organisation, the number of users at the organisation, the kinds of departments the organisation has, and the kinds of transactions the client organisation will need to carry out (Chau, 1995; Maiden & Ncube, 1998; Muscatello et al., 2003; Kato et al., 2003; Damsgaard & Karlsbjerg, 2010).

The live scenario is used in cases where it is decided that this provides the best option for showing the capabilities of a PS (Beecham et al., 2003; Jebreen et al., 2013b). In one of the case studies observed, the live scenario was chosen because the solution that needed to be demonstrated involved both hardware and software components. The live scenario aims to simulate a situation that could occur in the client company's real work environment. It was found, therefore, that contrary to simply describing or demonstrating a PS during a meeting, the live scenario may involve analysts actually creating a software demonstration environment that simulates the client company's site or their operations (Muscatello et al., 2003).

In the case studies conducted, it appeared that using a live scenario helped the analysts to better understand and respond to the needs of the client company. By conducting the live scenario the analysts were better able to see the challenges actually faced by the company. Previous studies have found that this kind of software demonstration can have a strong influence on whether a client will purchase a solution (El Emam & Madhavji, 1995; Chau, 1995; Maiden & Ncube, 1998; Muscatello et al., 2003; Kato et al., 2003; Al-Mashari, 2003; Haines, 2009).

It was also found that the planning of such a live scenario software demonstration relied on the ability of the team of analysts to use the report created by the sales and marketing team about the client's organisation structure and what initial issues were involved with creating any software required by the client organisation (Jantunen, 2010). This shows how the sales and marketing report ties in with the work done by analysts when planning software demonstrations, but it also suggests that the ability of analysts to plan an effective software demonstration or live scenario may to some extent depend on a) the quality, comprehensiveness, and clarity of the sales and marketing team's report, and b) the level of skills, ability, or knowledge that the analysts have acquired that may be needed for understanding the sales and marketing team's report (Jantunen, 2010). Once again, the use of a live scenario also requires analysts to possess or develop soft skills such as being able to deliver software demonstrations in a way that is both personable and persuasive in terms of the client's context, not merely to display their knowledge about the software.

In pre-implementation the scenarios used are not 'virtual' or hypothetical but rather are 'run-time' cases that show exactly how the software works in the specific environment. In pre-implementation the purpose of such live scenario use is not to find out the client's needs, but rather to help sell the product to the client and to reduce any client resistance to the product by 'proving' that the product can work in the appropriate environment. The use of the live scenario during software demonstration, therefore, appears to be a method that is particularly pertinent to pre-implementation.

7.3.3. Mechanisms of Scoping and Creating a Packaged Software Offer

The results of observations of and interviews within the two organisations revealed that analysts attempted to define the scope of the software during discussions with potential clients about their needs. Analysts believed it was important to carry out this scoping process early on since this would help them to construct a software offer, since such scoping would help everyone involved to maintain control of the time taken for implementation. Collecting such information not only provided analysts with details about what the new software needed to do, but also helped them to see what its limitations would be and what features or modifications would be unnecessary. This step therefore helped them significantly with implementing a PS that would suit the client – but that required the least degree of customisation.

It was also observed that the steps the software companies took related to software scoping were generally limited to finding out information about only the core requirements of the system or solution to be customised (Poba-Nzaou & Raymond, 2013). This was found to involve transactions issues and software output/input format issues (Dittrich et al., 2009); during the scoping process the analysts were not concerned with discovering detailed requirements (Poba-Nzaou & Raymond, 2013). Another aspect that was considered during the scoping process was the cost of implementing the software. Analysts needed to take into account what clients might be prepared to pay and what software was worthwhile for their own company to implement, before deciding on what software offer to make (Jebreen et al., 2013a). However, this has been noted as a challenge for SME companies; Poba-Nzaou & Raymond (2013) found that when SMEs entered the phase of planning the project "neither the budget nor the schedule of the next phases of the ERP adoption project were formally planned".

The study results also indicated many of the aspects that software companies took into consideration when making a software offer and the aspects of software and pre-conditions to be met that is mentioned in the software offer. It was found that when creating a packaged software offer, the software company decided on the scope of the offer and exactly how to develop the software based on the client's initial requirements, the modifications requested by the client, the extent of the modifications, and the technical requirements involved in meeting such requirements (Jebreen & Wellington, 2013a). Many of the core requirements considered during this process included transaction functions, software output forms, and technical dimensions of creating and running the software. These technical dimensions may take into account the client's infrastructure on site (Soh et al., 2000).

It was also found that the software companies used different kinds of assessment criteria when considering how to make a software offer. The assessment criteria were used to estimate the effort and time needed to develop, customise, and modify the packaged software (Jebreen & Wellington, 2013a). These assessment criteria related to various offer elements. The assessment criteria mentioned by the General Manager of one software organisation were New Features required, Customisation, Software Output/Input modification, and Technical needs. This process differs from processes in traditional RE as the software development company has some control over deciding the scope of the project, but must also take into consideration whether they are able to meet the client's requirements, may possibly have to decide between more than one solution they could offer the client, and may have to consider that they could be competing with another software company when making a software offer.

Analysts engaging in pre-implementation must think about the client's specific issues and decide whether any existing packages offered by their company can offer a solution. They will need to consider the time and cost involved with implementing a particular package or with making requested changes to that package, and they may decide to refuse a request for a particular solution if that solution falls outside the scope of the company or outside the scope of their current products. In this regard, pre-implementation considers how to deal with requests for modifications to existing functions from the perspective of managing the balance between software specificity and generality. Similarly, when an analyst engaging in pre-implementation receives requests for changes to software, they are concerned with this balance between specificity and generality, as modifications may disrupt standards in the software.

7.4. Packaged Software during Implementation

After packaged software offer is accepted by a client, analysts move on to arranging and conducting software implementation. In the implementation phase, analysts follow various practices, which include installation, software demonstration, identifying misalignments, and choosing how to respond to the discovery of misalignments by using various assessment criteria. Here, 'installation' refers to installing a copy of the packaged software prior to final implementation; 'software demonstration' refers to analysts demonstrating the software in the presence of users while also explaining the software's functionalities; and 'assessment criteria' refers to factors such as the genuineness of a misalignment, the size of the client company, and the software scope, which are all taken into account as analysts decide how to respond to misalignments that are identified.

7.4.1. Installation

The installation of a copy of the packaged software to identify technical misalignments between the packaged software's technical requirements and the users' IT infrastructure is one of the main implementation practices. Analysts may use the installation of a copy of the software as a way to discover the technical requirements for the software's implementation, or in order to gain a further in-depth understanding of the users' needs. In this way, software integration issues or problems with a company's infrastructure can be discovered prior to a real implementation of the software. It appears important that packaged software analysts run such checks prior to implementation because when implementing packaged software, there is a great need for certainty regarding whether what the packaged software requires is matched by what the users' IT infrastructure delivers. There are many kinds of issues and misalignments that could cause problems during implementation, such as insufficient server capability, insufficient speed, storage space, or RAM in computers used by the client, or server incompatibility. Clients' computers could also be missing necessary components or files. In the example below, taken from my day by day descriptive analysis, an analyst from Organisation 1 installs PS in order to identify technical misalignments:

*…… So the software was being installed **[installation]** to help the analysts assess the technical needs associated with the software, and in order for the analysts to discover any issues that could arise with the client's/ users' infrastructure **[identifying misalignments]**. If issues came up with the technical needs of the software (i.e. if the client did not currently meet those needs, or the client's technical provisions didn't match with the software), analysts would then assess the likely cost of making the software needs and the client's technical specifications match up **[Feb 12, 2012 - Organisation 1 – Installation]***

Analysts need to identify the misalignments between software technical requirements and users' infrastructure capability prior to implementation, and the primary means of doing so is through the installation of a copy of the software. It will not always be sufficient for analysts merely to question users at the client company about their infrastructure: in many cases, even the IT staff of the client company will not know enough about the infrastructure to be able to provide the needed information (Buonanno et al., 2005; Laukannen et al., 2007).

7.4.2. Software Demonstration

By using the installed copy of the software prior to official implementation, analysts can not only ‘test’ the software and how it runs in the client’s environment, but they can also educate users about the software’s functionalities and help their clients to create a vision about how they would use the software’s functions in their company. Through their interactions with a package, users can more easily identify the functions they need and desire, and define the functions that are not available in the package (Sia & Soh, 2007). Users who are shown a copy of software in action will gain a better understanding of what the software is able to do. Having seen the software actively working may also help to increase the degree to which users partake in discussions about the software with analysts (Beecham et al., 2003; Poba-Nzaou & Raymond, 2013). Here an analyst from Organisation 1explains the purpose of installing a copy of the software prior to final implementation:

Today I went along with the analysts while they travelled to the client’s site to actually install a copy of the software that had been created. By this point, the analysts had made a software offer to the client, and the client had accepted it. So, I watched the analysts trying to install the software. “We start explaining the software functions. So we install a copy of the software, not to be used in a real process, but to help us to explain our software functions to users..... I think that helps the users know what software can do.... As you know, users sometimes have no understanding of what software can do, so we use a copy of our software to teach them and to help users to engage in a good discussion about the software functions and their business process [User4]” ***[Feb 12, 2012 - Organisation 1 – Installation]***

Analysts explored the performance of the software by looking into issues that had been mentioned on reports dating from the pre-implementation stage. During such a demonstration, then, analysts may check for issues related to ‘transaction issues’ or ‘output/input format issues’ and look for ways to fix these issues without having to resort to customisation. If it is discovered that the software lacks a functionality or mechanism that is necessary to the client’s company, then customisation of the software will have to take place (Marbert et al., 2003; Snider et al., 2009; Poba-Nzaou & Raymond, 2013). For example, in one of my case studies, a transaction misalignment was found, which required customisation of the software. The client in question needed the software to feature a mechanism by which 80 JD could be deducted from the first month of an employee's salary and later returned to the employee after 3 months. In this case, the analysts minimised the customisation effort by explaining the benefits of the software when kept in its present form, but then agreeing to customise the software in terms of a transaction formula.

A software demonstration for packaged software is as a means of convincing users that there are alternative solutions to using customisations in response to misalignments (Khoo & Robey, 2011). The general recommendation from the analysts observed was that demonstrations of a trial version of the packaged software should be used as part of the implementation process to educate users about the software's functionalities, to discover and discuss user needs and misalignments, and to increase users' participation in discussions. Such recommendations support the findings of some previous research in this area/on the subject of (Al-Mashari & Al-Mudimigh, 2003; Khoo & Robey, 2011).

7.4.3. Identify Misalignments

The study found that misalignments can consist of the incompatibilities between packaged software and organizational in term of technical needs, transaction, and output/input. In the excerpts from my day by day analysis below, an analyst identify misalignments:

The main things that happened during the meeting today were that the analysts installed the software at client-server computers, and this related in their finding a range of problems related to how the software ran on the client's current infrastructure. These problems could be considered potential misalignments. Some of the issues that were discovered were that there were problems with the server compatibility: these problems related to speed, space, and RAM size. There were also problems on the client-server side. The client-server computer was missing some components that would be needed to run Dell files. The Team Leader explained to me about the problems with the client-server computer and the desktops at the client's organisation: "Everybody had to go and visit each of the desktops to install the apps ... the users' computers are not compatible with our software requirements so we have to fix them" [User5] ***[March 11, 2012 - Organisation 2 – Installation]***

The main actions taken by the analysts during the software demonstration/meeting included the following: in relation to customization requests, showing/confirming that there were no output and input change needs, discussing the client's business process in relation to how they provided staff uniforms and reimbursed staff for the uniforms' purchase or return, and a discussion of the possibility of changing the transaction formula related to their provision of uniforms ***[Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]***

In the case of this HRMS software, several technical issues were discovered by installing the copy of the software. For example, issues were found that related to server compatibility, such as speed, storage space, and RAM size. Other issues were found on the users' desktop side, such as their computer missing some components that were related to running files.

After the software was installed successfully, analysts used the version of the software to carry on identifying misalignments by following issues mentioned on the report from the pre-implementation stage. For example, in the case of this HRMS software, the users' issues were categorised under 'transaction issues' such as 'add employees', 'bonuses mechanism', 'payments made for uniforms' and 'output format issues' such as the software reports format. In this case relating to HRMS, a transaction misalignment was found, which required customisation of the software. It is important to note, following Haines (2009), that the "customization" notion is not employed in a uniform manner within the literature. In fact, this author observes three distinct types of software customisation, namely configuration, extension (i.e., through user-exits), and modification of the source code. Configuration is done through software switch-setting by changing entries in configuration tables, files, or editable rules.

Wei et al. (2005) described misalignments as arising "from company-specific, sector-specific, or country-specific requirements that an ERP package does not support and can be clustered into data, process, and output". Soh et al. (2000) defined misalignment as "the incompatibilities between organizational requirements and ERP software in terms of data, process, and output", or in terms of "opposing structural forces between an ERP system and the implementing organization".

Meanwhile, Sia & Soh (2007) also identify four misalignment types. These include: Imposed-Deep, Imposed-Surface, Voluntary-Deep, and Voluntary-Surface. Imposed-Deep refers to there being a missing or inappropriate thing, property, state, or transformation in the system, arising from different country or industry assumptions. Imposed-Surface relates to such things as missing or inappropriate access, input, presentation, or output in the system, and can arise from different country or industry assumptions. Voluntary-Deep relates to there being a missing or inappropriate thing, property, state, or transformation arising from organisation-specific assumptions. Lastly, Voluntary-Surface relates to there being missing or inappropriate access, input, presentation, or output arising from organisation-specific assumptions.

7.4.4. Assessment Criteria

The study found that analysts respond to the discovery of a misalignment in one of two ways. They will either suggest that the user's company adopt the packaged software as it is and make the most of its current functionality, or they will agree that the packaged software needs modification in terms of customising a function or adding a new feature. If the client company is told to adopt the software as it is, doing so might require the company to change their business processes. It was found that analysts take a number of factors into account before deciding what action to take in response to finding a misalignment.

The first thing that analysts determine is whether any misalignment discovered is an 'actual' misalignment, or only a perceived one (Van Beijsterveld, 2006). A misalignment is genuine and only truly requires modification if the software does not support such a transaction or transaction formula as required by users (Sia & Soh, 2007). It was found that misalignments sometimes seemed to exist in cases where the software's functions did support a particular desired transaction, but not in the order or with the kind of interface that was expected (see also Sia & Soh, 2007). In such a case, the analysts in the case studies did not consider the misalignment to be 'actual' or genuine, only to be a 'perceived' misalignment. In the excerpt from my day by day analysis below, an analyst explains how his team views the difference between perceived misalignments and actual misalignments:

"Yesterday we met with users for HR software, and they kept stressing that the software did not have this function or this function, and so on. Actually, our software does provide those functions, but not in the order they want, so in such a case we explain to users how such functions work .... However, if the request from the user is really something that does not exist in the software, such as a transaction formula, and if it requires us to make new features, then we have to make them" [User5] ***[Feb 27, 2012 - Organisation 2/ Respond to the discovery of a misalignment]***

The second thing that analysts determine is whether the request to fix the misalignment is within the software scope or outside the software scope (they determine this by referring back to the original software offer). They also consider the size of the users' organisation and their probable ability to pay for the required customisation (Haines, 2009). The following excerpt from my day by day analysis shows analysts making such methodical assessments:

The second main consideration I saw the analysts making was whether each customisation request was within the software scope, or outside of the software scope. They explained that here, the 'software scope' means whatever functionalities or features the analysts had agreed to provide in the software, as detailed in the original software offer that had been accepted. If the client was now asking for functions or features that had never been mentioned in that software offer, then they were going beyond the scope of the software offer ***[Feb 27, 2012 - Organisation 2/ Respond to the discovery of a misalignment]****.*

It was found that in addition to these three main factors that act as assessment criteria for deciding how to respond to requests about misalignments, one other factor internal to the software development companies affected their decisions about customisation. Both software development companies observed during the ethnographic study had official policies of aiming to minimise the customisation of software as much as possible. Previous research has shown that during initial implementation, most organisations will customise software (Newman & Zhao, 2008; Zach et al., 2012). There are repercussions to engaging in customisation, however – as Khoo et al. (2011) observed, customisation is usually associated with increased costs and longer implementation time "Customisation of packaged software is often not a trivial activity, as it can involve several hours – or even months – and can involve a substantial cost. It is therefore essential that companies provide value through their packaged software by placing an emphasis on recognizing their client's requirements of the software being implemented" (Khoo et al., 2011). Because of the difficulties and extra effort associated with modification of software, analysts taking part in the ethnographic study sometimes expressed their reluctance to engage in customisations unless they were totally necessary:

The Team Leader told me that "We have to reduce making changes [customization/new features] in our software. Cause it's not easy to do. It's not easy to throw away software that has been developed over several years ***[Feb 27, 2012 - Organisation 2/ Respond to the discovery of a misalignment]****.*

One final finding relating to misalignments between software and clients' needs was that the discovery of a misalignment was not always seen by the analysts and software companies as a bad thing. The software companies could in fact derive some benefits from such discoveries as they could provide them with ideas for improving their current packaged software and for adding functionalities to future releases. One analyst from Organisation 1 expresses such a viewpoint in the excerpt below:

> *Some users ask us to change our software functions or to add new features. Anyway, in this case some of these requests could be useful for our software so that we can improve the software functionality, so we do it even if it's outside our scope. But still, assessment of customisation risk is important, so maybe we can't add the function to the package that the current user wants because its development needs a lot of time, would cost a lot to do, and is outside the scope. But if we consider these features as important to add to our software, maybe these features will be in our next release [User4]* ***[April 4, 2012 - Organisation 1/ Benefits of misalignments].***

7.5. A Parallel Star Model for PSIRE

This PhD delivers an understanding of PSIRE that represents how SPSVs might identify and respond to misalignments between users' needs and PS functionalities. It does so through proposing a Parallel Star Model for PSIRE which is based on empirical observations made during my ethnographic research. These findings regarding the Parallel Star Model for PSIRE were drawn primarily from the day by day descriptive analysis forms and results presented in Chapter 5, 6 and this chapter. Furthermore, I found that the structure of Hartson & Hix (1989) model in their study of human-computer interface development is appropriate to my analysis and so utilised this in forming up the Parallel Star Model for PSIRE. Beyond that, however, the context, being PSIRE, and content, in terms of the actions undertaken, are clearly distinct.

The model by Hartson & Hix features a star-shaped diagram showing the life cycle of human-computer interface development, with the "evaluation" of five different processes located at the centre of the cycle. The main similarity between Hartson & Hix's model and my own lies in the use of a star-shaped configuration to show the possible interconnections between different processes involved in the development and provision of packaged software (or in Hartson & Hix's study, the development of interfaces). Both models feature a group of processes that are connected to each other by means of a central step which relates to making assessments about the next action to take. In Hartson & Hix's (1989) model, this central step is "usability evaluation"; in my model, the central step is "assessment".

The Parallel Star Model is designed to depict the parallel nature of the processes conducted during PSIRE: feasibility study, assessment, implementation, software demonstration, and identifying misalignments. I have noted that analysts conducting PSIRE are commonly able to carry out multiple processes at the same time. Therefore, I hypothesise that PSIRE is actually conducted in terms of a parallel process model rather than a linear process model. Here, 'parallel' means that analysts can conduct a number of processes at the same time. Furthermore, I have noted that analysts before conducting another processes/engaging in another group of multiple actions, the analyst moves back to a central position of assessing the needs of the packaged software to be implemented. Further, proposing a Parallel Star Model is appropriate in that these processes can happen at the same time and in *no particular order*. An instance in the data reported below illustrates how the Parallel Star Model fits here, in that an analyst from Organisation 1 takes one action, installing the PS, in order to carry out multiple tasks: to identify technical misalignments, to assess misalignments, and to conduct a feasibility study.

…… So the software was being installed ***[installation]*** *to help the analysts assess the technical needs associated with the software* ***[assessment]****, and in order for the analysts to discover any issues that could arise with the client's/ users' infrastructure* ***[identifying misalignments]****. If issues came up with the technical needs of the software (i.e. if the client did not currently meet those needs, or the client's technical provisions didn't match with the software), analysts would then assess the likely cost of making the software needs and the client's technical specifications match up* ***[assessment & feasibility study] [Feb 12, 2012 - Organisation 1 – Installation]***

As shown in my example from Organisation One, the Parallel Star model suitably demonstrates that several (groups of) processes can be carried out in parallel by analysts during the practice of PSIRE. In the Parallel Star Model, not only can processes be carried out in parallel, but analysts can readily move back and forth between different processes once they wish to move to a new activity or complete the parallel processes in which they were engaged. We can see from the excerpt above that there is a parallel process to installing the PS and identifying technical misalignments, and that the assessment of technical misalignments involves following a ‘star’ process. The PSIRE process can be said to follow a ‘star’ model because after each individual process or group of processes is followed, an analyst usually takes a ‘central step’ of stopping to assess the various dimensions of other misalignments. My findings in this regard support previous findings by Poba-Nzaou & Raymond (2013) who argue that their study confirms that “flexibility is what SMEs seek most in an ERP system, not the “best practices” embedded within these systems when they are pre-packaged”. An SME “can proceed in a rather intuitive and unstructured manner” (Poba-Nzaou & Raymond, 2013; Jebreen et al., 2013b).

The Parallel Star Model for PSIRE (shown in Figure 7.5-1) uses a star-shaped configuration to show the possible interconnections between the different processes involved in the implementing and provision of packaged software. The model features processes that are connected to each other by means of a central step which relates to making assessments about the next action to take or activity to engage in. In abstract terms, this central step involves the analysts pausing to check information and to carefully consider the next step. In more practical and specific terms, the central step comprises various forms of “assessment” involved in PSIRE in order to support analysts making decisions related to misalignments that have been found between the packaged software and the client’s requirements or the client’s business environment.

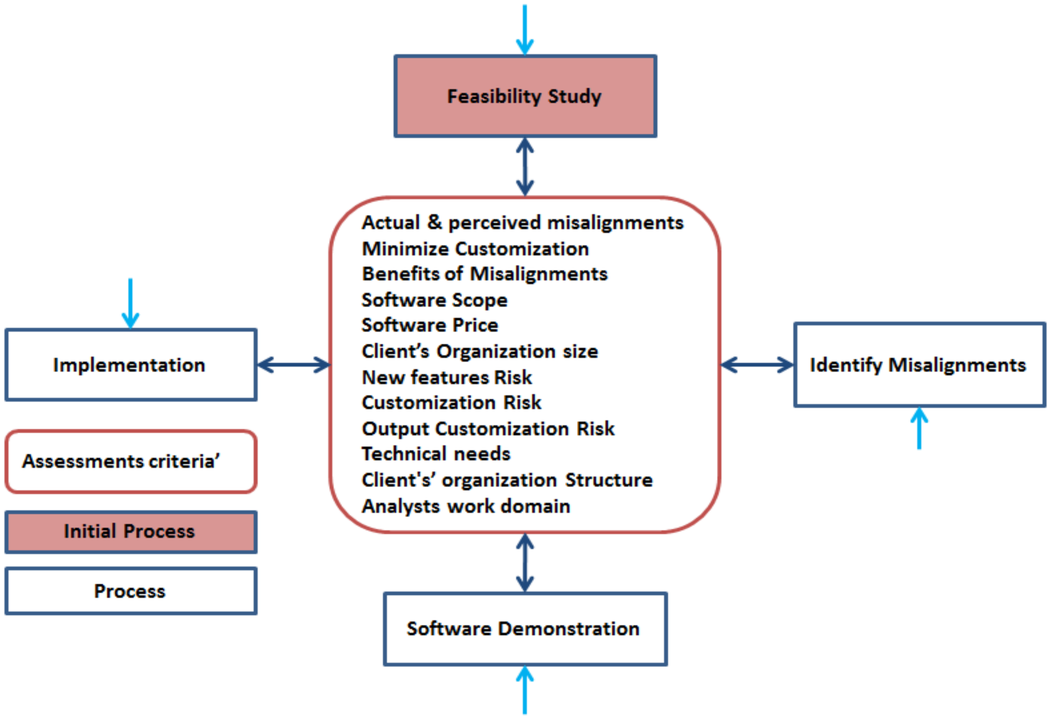


Figure 7.5-1 PSIRE Parallel Star Model

In work related to that conducted here, Poba-Nzaou & Raymond (2013) note that "at Bio-Epsilon [an SME], the consideration of the risk of implementation from the adoption stage was based on a reactive, informal, intuitive, and incremental approach". Their approach to risk management was thus guided by some specific principles and policies, but the owner-manager admitted that the process was informal and that he had been guided by events. Poba-Nzaou & Raymond (2013) remark that this approach is in contrast with the 'ideal' ERP adoption approach prescribed for SMEs in prior literature (Verville et al., 2005). They further cite numerous previous studies that indicate between a quarter and a half of all SME companies adopting an ERP system may not conduct any formal evaluation (Marbert et al., 2000; Olhager & Selldin, 2003). Other studies have shown that SMEs frequently customise the ERP software they adopt (Marbert et al., 2003; Snider et al., 2009). All of these considerations lead to a different kind of RE life cycle approach in the Parallel Star Model. This model shows that during

PSIRE, processes can be carried out in a parallel star approach that allows for relative flexibility; they do not always have to be followed in a particular order.

In the Parallel Star Model, the misalignments found may relate to output/input functions and to the user interface, but more commonly relate to transactions (at least in the field study sites considered here). While engaging in "assessment", analysts need to consider both the software dimensions and business dimensions of responding to misalignments. In terms of the software dimensions, there could be risks to ongoing support of the software if modifications are made. In terms of the business dimensions, the analysts will consider whether dealing with the misalignment is within their work domain, and whether there is any benefit to their organisation from dealing with the misalignment. The Parallel Star Model, however, shows not only that the processes involved in PSIRE are interconnected in flexible ways, arranged around the central step of "assessment", but that multiple PSIRE processes can be enacted simultaneously. An analyst from Organisation 2 worked on identifying misalignments by software demonstration, which also helped with the assessment of misalignments:

Today I accompanied the analysts on a trip to the client's company site. The analysts engaged in a software demonstration of the HR software they had developed for the client. The client had already accepted the software offer made by Organization 2 ***[feasibility study]****, but now a more in-depth understanding of the users' needs was required. I observed that there seemed to be various reasons behind the meeting/demonstration: some were to do with the software itself, some were to do with the client's requirements, and some were to do with the client's company environment. As the analysts showed the client the software and discussed the client's needs from the software, it was apparent that the analysts were trying to validate the client's needs (i.e. confirm those needs/requirements they had already collected), to get a better understanding of the client's business process, and to identify any mismatches between the client's needs and the software* ***[software demonstration]****. These actions helped the analysts know more about the client's requirements* ***[identifying misalignments]****. While showing the client the software, the analysts also engaged in explaining different elements of the software. They explained its various functionalities and clarified various textual and graphical materials. The analysts tried to demonstrate the software in a way that showed how it could support the client's business. They also trained some users at the client's organization on using some of the software functions* ***[software demonstration]****……. By showing the client all of these software functions and drawing diagrams of the client's user environment the analysts were able to get more information about the necessary software scope, the client's business process, and any mismatches remaining between the software and the client's business practice* ***[identifying misalignments]****……The main actions taken by the analysts during the software demonstration/meeting included the following: in relation to customization requests, showing/confirming that there were no output and input change needs, discussing the client's business process in relation to how they provided staff uniforms and reimbursed staff for the uniforms' purchase or return, and a discussion of the possibility of changing the transaction formula related to their provision of uniforms, and using a printed screenshot of the software to validate the client's needs; explaining the report list provided by the software* ***[assessment]****; training users on other functions of the*

HR software. The analysts ended up signing off on the 'change request' that the clients had made regarding the formula related to uniforms. To me, it looked like there were two main benefits to having such a 'change request' form and needing to have it signed: it would be useful to have an official documented request to refer back to later ***[assessment]****, for information or for confirmation that an agreement was reached, and this request form helped to manage the whole implementation process……The analysts did turn down one customization change request, though. The Accounting Manager of the client organization then asked the analysts to add some attributes to the employee' salary reports* ***[assessment]****. The analysts answered this by saying that the attributes that were being requested already existed within the software. At the end of the meeting, the analysts signed off the users' meeting summary. had a list of topics to discuss during the meeting, or a list of needs they wanted met, and when the analysts signed off, it meant they were saying they'd spoken about all those topics or agreed with all those requests* ***[Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]****.*

Because multiple processes can be carried out at the same time or swapped between quite easily, one particular benefit of the model (and the approach represented in it) is that it reduces the ordering constraints acting upon process activities. For example, analysts do not necessarily have to have found all of the misalignments that are present before working on the training of users. In fact, analysts can train users in how to utilise the software at the same time as they identify misalignments. They can also validate that they have changed the software to deal with a misalignment at the same time as training the users, or can identify technical misalignments in parallel with software installation. Analysts can also move back and forth between finding misalignments and developing a solution to the misalignments, and looking for more misalignments. An analyst from Organisation 1 demonstrates their use of the software to identify misalignments, and in their assessment of misalignments, in the following:

…… the purpose of the meeting (as explained to me by the analysts) was to demonstrate the functions of the software to the client, to educate the client about the functionalities, to validate the needs they already believed the client had, and then to collect more information about what the client needed ***[Software demonstration]****. The software demonstration also offered a chance to discover if there were any issues with the client's current business process that could cause a software-client mismatch* ***[identifying misalignments]****. The lead analyst began to demonstrate the software's functions to the client, always trying to show how they related to the client's business process. There was also some general discussion about the client's business process* ***[Software demonstration]****. It was soon discovered that there was a problem relating to the software lacking some function attributes that were needed* ***[identifying misalignments]****. These attributes were particularly important. The client explained that they needed software that could allow the student to fill out information related to their family's medical history. While the software currently demonstrated provided the possibility of filling out basic information about the students (like contact details, date of birth, and so on) and about the students' general medical issues, it did not provide any field that let students fill out information about their family's medical history* ***[identifying misalignments]****. This was a problem as the clinic needed to know if there was a history of diabetes in the students' family, or a history of cancer, and so on. The analyst and the user therefore agreed that a modification should be made to the attributes of the function relating to inputting students' information* ***[assessment]****. Another field will be added to the function so that family medical history can*

be recorded. As part of the process of collecting more details about the user's needs and business process, the analyst also collected the user's document forms that detailed what attributes they needed different functions to have. The analyst and client then discussed the information that the client had provided on the document forms. The analyst looked at the needs that the client had listed on their document forms, and explained how the current functions in the software provided the attributes that were needed ***[Software demonstration]****. If necessary attributes were missing, they wrote this down* ***[identifying misalignments & assessment]****. The client decided to make a modification request regarding the attribute needed to collect family medical history, so modification request documents were filled out* ***[March 12, 2012 – Organisation 1 - During Implementation – Software demonstration & identify users' needs]***

Examples of such multiple processes being carried out simultaneously are depicted in Figure 7.5-2 and Figure 7.5-3. It is essential to support continual assessment and iteration during PSIRE, including small loops of iteration; the Parallel Star Model supports such an approach. For example, as a software demonstration is carried out, at the same time, analysts can identify misalignments. It is possible to carry out training of the users and identify misalignments at the same time. It is also possible to carry out training of the users at the same time as validating misalignments: after the analyst has developed software in a way to fix a misalignment that was identified, they can show their new development to the user, thus validating that the misalignment was fixed while simultaneously training the user. An analyst from Organisation 1 trained users at the same time as validating misalignments:

Today I attended another meeting between an analyst from Organisation 1 and the client representative from the university library. The meeting followed on from the prior meeting on March 13 at which the analyst had agreed to make some modifications to the library software. The first thing the analyst did was to show the client the list of what modifications had been agreed to at the last meeting ***[software demonstration]****. Then they started explaining what modifications they had made to the software functions. Their next task was to demonstrate the software to actually show that these modifications had been made successfully and that they reflected the client's requests, and to show the software actually working* ***[software demonstration - validating]****. The analyst then began to install the software copy at the client's site. While installing the software copy they actually upgraded it as they went, by adding in the modifications requested* ***[installation]****. They then imported the Excel data spread-sheet of student information into the software database. Once done, this meant that the software was ready to be used in a live scenario demonstration* ***[software demonstration]****. The analyst then carried out a software demonstration on their own to validate that the modifications had been made and to show how the software functions worked* ***[software demonstration – validating - training]****. Then, as part of a live scenario demonstration, they got the client to use the software as though they were issuing books, and trained them on how to use the software functions* ***[software demonstration – training]****. The client was happy with everything that had been done and with the training they had received, so at the end of the meeting the client signed the modifications acceptance form and the trained functions list. These two documents are drawn up by the software company in order to keep proper documentation of the implementation process and to have proof that the software was officially accepted* ***[March 21, 2012 - Organisation 1 - During Implementation – Software demonstration & validation of users' needs]***

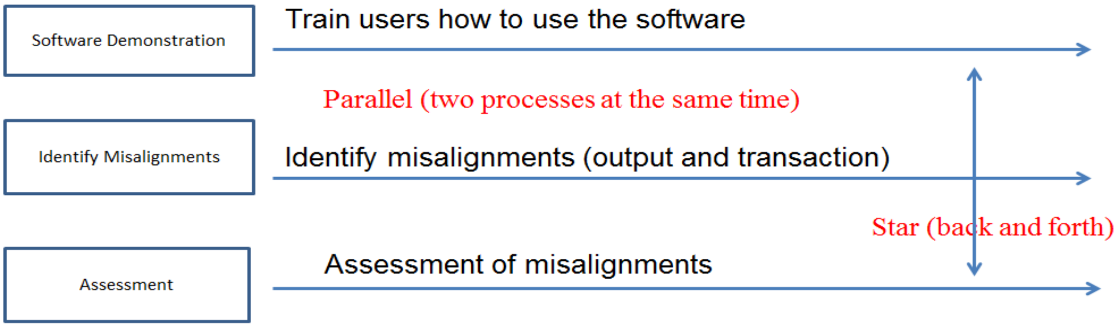

Software Demonstration
Identify Misalignments
Assessment
Train users how to use the software
Parallel (two processes at the same time)
Identify misalignments (output and transaction)
Star (back and forth)
Assessment of misalignments

Figure 7.5-2 Parallel Star Process example 1

In other examples, analysts from Organisations 1 & 2 installed the PS in order to identify technical misalignments, and to assess those misalignments:

> *The Team Leader also mentioned that this kind of installation was useful because it was more effective than just trying to ask an organisation's IT people about their organisation's infrastructure: "We cannot ask users about their infrastructure because most of the users are not IT people and even IT people don't know some of the infrastructure requirements for software to be run … We try to discover any issues with the users' infrastructure, but we do that by installing a copy of our software … that's the only way to get to know the issues with the users' infrastructure" [Team Leader]* ***[installation]****. The main things that happened during the meeting today were that the analysts installed the software at client-server computers* ***[installation],*** *and this related in their finding a range of problems related to how the software ran on the client's current infrastructure. These problems could be considered potential misalignments. Some of the issues that were discovered were that there were problems with the server compatibility: these problems related to speed, space, and RAM size* ***[identifying misalignments]****. There were also problems on the client-server side. The client-server computer was missing some components that would be needed to run Dell files* ***[identifying misalignments]****. The Team Leader explained to me about the problems with the client-server computer and the desktops at the client's organisation: "Everybody had to go and visit each of the desktops to install the apps … the users' computers are not compatible with our software requirements so we have to fix them"* ***[assessment]****. [Team Leader][****March 11, 2012 - Organisation 2 – Installation****]*
>
> *…… So the software was being installed* ***[installation]*** *to help the analysts assess the technical needs associated with the software* ***[assessment]****, and in order for the analysts to discover any issues that could arise with the client's/ users' infrastructure* ***[identifying misalignments]****. If issues came up with the technical needs of the software (i.e. if the client did not currently meet those needs, or the client's technical provisions didn't match with the software), analysts would then assess the likely cost of making the software needs and the client's technical specifications match up* ***[assessment & feasibility study] [Feb 12, 2012 - Organisation 1 – Installation]***

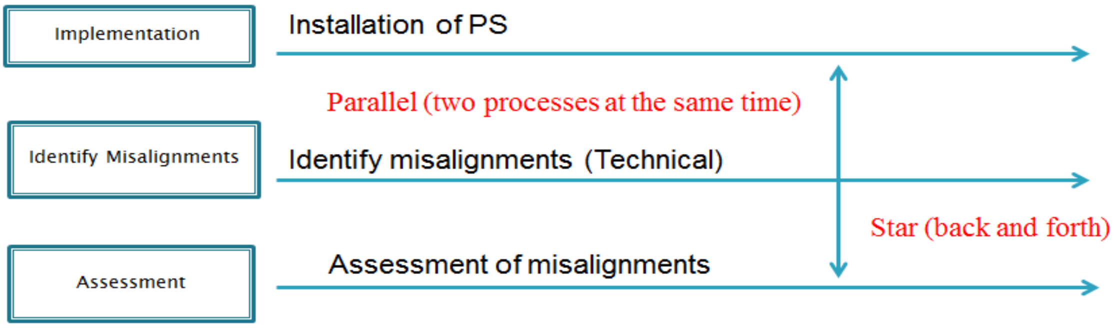

Implementation
Installation of PS
Parallel (two processes at the same time)
Identify Misalignments
Identify misalignments (Technical)
Star (back and forth)
Assessment
Assessment of misalignments

Figure 7.5-3 Parallel Star Process example 2

Conventional life cycle models lean toward independent performance of each process and impose limitations on the sequence of the processes. However, the Parallel Star Model for PSIRE supports the conduct of interdependent, interwoven processes. The Parallel Star Model is flexible, as there are very few constraints involved: the only major limitation on analysts is that they will usually need to go through the central assessment process before moving on to beginning a new process.

As shown in Figure 7.5-1, the form of assessment focusing on the software dimension of implementing packaged software in response to misalignments involves addressing the risk of adding new features, the risk of customisation, the output customisation risk, and the technical needs of dealing with the misalignment. Analysts will consider whether they can or should carry out all of the modifications desired by the user, and what technical risks or risks to the software would be involved in carrying out such modifications. They will assess, for example, whether the changes made would have significant impact on the software functions, and especially whether they would disrupt essential functions. They will also assess whether the software may be disrupted even if a non-essential (‘redundant’) function is modified or removed; as illustrated in the following:

Today I had a discussion with the Team Leader of the analysts from Organization 2 about the software demonstration they had done for their client the day before (the demonstration of HR software). The Team Leader said a few things that helped me understand how the analysts make decisions about how to respond to customisation requests from a client. The analysts have to decide whether it is worthwhile, cost-effective, or even possible to fix the misalignment between the client's needs and what the software provides, or not ***[assessment]****. From the Team Leader's comments I also realised that analysts make a distinction between whether a misalignment is an ‘actual’ (real) misalignment, or whether the clients* *only perceive it as such – i.e. sometimes a misalignment only apparently exists – from the client's point of view – because the client thinks that the software cannot do a particular function* ***[assessment]****, whereas the analysts might be aware that the software actually can provide that function. The Team Leader told me that "We have to reduce making changes [customization/new features] in our software. Cause it's not easy to do. It's not easy to throw away software that has been developed over several years* ***[assessment]****...... The decision of whether they should make the customization/ new feature or not seems to be guided by two considerations: first, the analysts decide whether the particular misalignment that was discovered is actually a real misalignment or not. I witnessed the analysts making a distinction between 'actual' misalignments and misalignments that were only 'perceived' by the client* ***[assessment]****. The second main consideration I saw the analysts making was whether each customisation request was within the software scope, or outside of the software scope. They explained that here, the 'software scope' means whatever functionalities or features the analysts had agreed to provide in the software, as detailed in the original software offer that had been accepted* ***[assessment]****.....From my general interaction with this Organisation (and also with Organisation One) I can see that these two medium-sized software companies don't really want to carry out many customisations. Both organisations have a policy of minimising customisation as much as possible. This explains why the analysts from Organisation 2 were quick to show their client that their software actually did provide one of the transactions that they wanted. I've concluded that these medium-sized software companies want to minimise customisation as much as possible, and that when they consider whether to go ahead with a customisation, they are less likely to do it if a) the customisation request is outside of the software scope, and b) the misalignment that the client identified is only a 'perceived' misalignment.* ***[Feb 27, 2012 - Organisation 2/ Respond to the discovery of a misalignment]***

Meanwhile, Figure 7.5-1 also displays the form of assessment dealing with the business dimension of making changes to the software. This form of assessment involves addressing whether misalignments are actual or perceived, making assessments related to the preference to minimise customisation, considering the client organisation's size, considering the software scope and the software price, and addressing the possible benefits to be gained from working with misalignments. The first consideration they make is whether a misalignment that has been discovered is an 'actual' misalignment, or only a 'perceived' one. An 'actual' misalignment would mean that the software does not support a desired transaction or its transaction formula. A misalignment is 'perceived' rather than 'actual' when software functions do support a particular desired transaction, but in a way different from that expected by the users. Even when the misalignment that has been found is 'actual', analysts will still stop to determine whether the misalignment is within or beyond the software scope. Here, 'scope' is determined by looking back at the original software offer that the analyst company made to the clients during the pre-implementation. The size of the client's organisation and the price they are willing to pay for software or for customisations is also an issue that is considered during 'assessment'. When considering whether to go ahead with customisations, analysts consider the size of the user's organisation, because larger organisations can generally better afford customisations.

It has been noted that the Parallel Star Model has few constraints as to the sequence in which processes can be followed. That said, the feasibility study and the installation process are the initial processes needed to set up the software environment. This is an initial constraint on the model; these two actions must be taken before analysts can use conduct other activities. Note here how the feasibility study is conducted before the next action, in Organisations 1 and 2:

I accompanied analysts from Organisation 1 to a software demonstration at the site of a client who needs new software for their medical clinic. The client provides medical services for a school needs to be able to record student details and medical history. Two analysts attended the meeting with the client. The client had already accepted a software offer from Organisation 1 ***[feasibility study] [[March 12, 2012 – Organisation 1 - During Implementation – Software demonstration & identify users' needs]***
Today I accompanied the analysts on a trip to the client's company site. The analysts engaged in a software demonstration of the HR software they had developed for the client. The client had already accepted the software offer made by Organization 2 [feasibility study] ***[Feb 26, 2012 - Organisation 2 – Software demonstration, identify users' needs, validation of users' needs]***

The installation process is a further initial process:

..... Installing the software in this way also gives the analysts a chance to educate the client and their users about the software product's functionalities [installation]. When I discussed this with a few different analysts, the analysts told me that installing the copy was a way to educate users about the software's functionalities, to help users create a vision of the software and how it could function for them, and to increase the users' participation in discussions ***[March 11, 2012 - Organisation 2 – Installation]***

7.6. Conclusion

The research reported in this study is one of few empirical studies focused on requirement engineering practices for packaged software implementation. It offers an in-depth, qualitative view of requirements engineering to implement packaged software. Packaged software is a unique type of IS software; packaged software engineering no longer involves building systems from scratch, but rather integrating existing frameworks and modules and responding to client requirements and modifications. Traditional software engineering has a group of influential approaches that are often considered good practices. However, these assumptions do not apply for PS implementation; PS implementation requires its own set of good practices.

Given the growing importance of packaged software and the apparent inevitability of packaged software implementation, it is increasingly necessary to understand the requirements engineering practices for packaged software implementation. My contribution to this effort is a comprehensive set and description of PSIRE practices, with a particular emphasis on PSIRE as it relates to SPSVs. This research draws its inspiration from earlier literature on requirements engineering and ethnographic settings. However, it extends earlier findings by drawing on data obtained during a new extensive ethnographic study.

Throughout this study I have attempted to demonstrate roles performed by analysts in PSIRE, and also how the nature of an SPSV affects the processes involved in PSIRE. In order to better understand such influence of the nature and organisation of SPSVs on PSIRE, I conducted an ethnographic study of two software development companies who create and implement packaged software. The ethnographic study was approached with several Research Questions in mind: three questions relating to analysts' practices in SPSVs.

This study provides a new understanding of packaged software implementation by carefully discussing the PSIRE practices and explaining why these practices occur; by shifting the study perspective used from one focusing on the concerns of the client companies during PS implementation to focusing instead on approaches taken by and activities inside the SPSVs who are the providers of the PS; by explaining the importance of various stages of PS implementation and analysts activities during implementation such as software demonstrations, use of live scenarios, mechanisms of scoping and elements considered when creating a packaged software offer; by offering information and analysis that should help SPSVs to conduct a feasibility study for PS implementation; and by offering and explaining a Parallel Star Model for PSIRE that has few constraints and that is the model to demonstrate how packaged software RE processes can be carried out in parallel. The Parallel Star Model is based on observations made during my ethnographic research into the RE practices at two SPSVs and is designed to support the processes involved in PSIRE.

However, it is now the case that in most organisations, new software is created by integrating functionality from existing software and components or by implementing packaged software. In such cases, it makes little sense to specify requirements in terms of what the software should do – the functionality is already defined in the software (Sommerville et al., 2012). Rather, I argue that requirements engineering practices for PS implementation should be approached from a misalignments perspective, which focuses on what functions the software provides, who needs a particular function in order to do their job, and what misalignments exist between software functions and users' needs. "There is a need for additional elicitation techniques not really covered by bespoke RE practices [for packaged software]" (Gorschek et al., 2011).

Chapter 8 Conclusion

8.1. Summary of the Study

This thesis provides empirically grounded novel insights relating to the understanding of the requirements engineering activities conducted for Packaged Software Implementation (PSI) as carried out in small packaged software vendors in Jordan. The research was motivated in part by the assertion by Merten et al. (2011) that SPSVs should be considered to be fundamentally different from large companies, and that most current requirements engineering practices are unsuitable for SPSVs (Jantunen, 2010; Quispe et al., 2010). It also responds to the fact that the literature on PS implementation has seldom featured discussions about SMEs (Zach et al., 2012). Poba-Nzaou & Raymond (2013) discuss the major trends in the history of the study of the adoption of packaged software and ERP systems and the study of the factors behind their customisation. They note that "the packaged software process has been studied principally with regard to its procedural characteristics and critical success factors" and that "for ERP packages in particular it has been found that the adoption process is strongly influenced by the social context, thus countering the technologically deterministic discourse that has generally surrounded such software". They believe that "a purely rational or deterministic view of the packaged software adoption process" is inadequate in SMEs.

I began research for this thesis by aiming to answer the question "How is requirements engineering in packaged software implementation contexts enacted at SPSVs?" My efforts to answer this question involved my examining past literature on PSI and bespoke software, and RE for SPSVs. Summaries of such literature, their theory and their findings, is presented in Chapter 2. However, the most appropriate way to answer this question was to study the SPSVs on site. Through an ethnographic study I was able to answer the question "What are the analysts' practices in the context of packaged software implementation by small packaged software vendors in Jordan?" but also, either through observation or through interviews with and surveys of analysts to discover many of the reasons why analysts used particular practices.

The quest to answer further major research questions led me to engage in ethnographic research in two small packaged software vendors in Jordan. The field work I conducted and my subsequent travel back to participants to validate the results assisted me in answering the questions: "How do analysts conduct these practices in the context of packaged software implementation by small packaged software vendors in Jordan? And "Why do analysts conduct these practices in the context of packaged software implementation by small packaged software vendors in Jordan?"

8.2. PSIRE Processes and Practices

This study presents an account of packaged software implementation requirements engineering with a focus on analysts' practices within SPSVs. Through this thesis I offer extensive discussion of how small packaged software vendors in Jordan approach the challenges encountered when conducting the implementation of packaged software at SMEs. I have discussed the different factors involved with finding clients, eliciting requirements and identifying misalignments, creating a packaged software offer, and modifying or customizing existing packaged software. My ethnographic account delivers an explanation of the day-to-day practices followed by analysts within small packaged software vendors in Jordan.

This section draws a number of conclusions from the results of the study as presented in Chapters 5 to 7.

1. The software analysts received software demonstration requests from their sales and marketing teams. The information these request forms tended to contain, along with that on the customer analysis forms, impact on how the analysts' team approaches demonstration presentations. The main factors affecting decisions on the presentations related to the client's organisation size; the number of users and departments within the client's organisation; the

transactions involved and the relations between the transactions; the number of database records; the organisation's level of security; the issues the client needs solved; the limitation of the work domain of the analysts' company; and the limitation of the software product solutions that the analysts can offer.

2. During pre-implementation analysts approached a software demonstration by considering two different dimensions: the business dimension, which consists of offering a solution to the client, and the software dimension, which considers whether it is possible to meet the client's initial requirements and new requirements. The analysts were actively involved in the pre-sale of their software and it was they who carried out the software demonstrations, not the sales teams of the SPSVs. An analyst involved with packaged software is required to take on more of a sales role; their aim during the software demonstration pre-implementation is to both identify the client's needs and to find any misalignments that are present, and to sell the software product.
3. The analysts were very aware of time constraints during software demonstrations and that they did not wish to unduly tire clients. Their objective during demonstrations, therefore, was to focus mostly on the client's key issues and how their software could deal with those issues, rather than on explaining every relevant function of the software. This meant that during software demonstrations, they tended to focus on the 'business dimension' of what they could offer the client organisation, rather than engaging in any extended technical software analysis.
4. The analysts at SPSVs often carry out software demonstrations by creating a live scenario through a real case. In doing so analysts may be better able to demonstrate the capabilities of a system or a software solution, how its software and hardware components work together, and how it can solve the client's issues. The live scenario can also be useful for collecting initial requirements.
5. Scoping mechanisms are used in order to prevent the implementation time and cost from expanding and thus threatening the project. The main considerations when creating a packaged software offer were the number of modifications

requested by the client, the extent of the modifications, and the technical requirements involved. The creation of a packaged software offer provides the first discussion of the theory related to ‘assessment criteria’ used by analysts. The assessment criteria discussed here include the ‘New Features’ requested, which assesses the number and extent of proposed new features for the existing package, ‘Customisation’, which assesses the impact that may result from modifying existing functions, and the ‘Software Output/Input’, which consists of creating new reports or modifying existing reports.

6. The installation of packaged software at the client’s site is a particular mechanism analysts used to identify technical misalignments between the packaged software and the client’s IT infrastructure. In fact, the need for certainty regarding whether the infrastructure the packaged software requires matches the user’s IT infrastructure is particularly important. Other forms of misalignments may also be found, such as output issues or problems with transactions. Furthermore, the installation is used to carry out a software demonstration. Analysts mentioned that in such cases, they did this in order to show the users the software’s functionalities and to increase users’ participation in discussions.
7. Packaged software companies determine whether misalignments are real or only ‘perceived’ and the responses that they make to clients follow as a result. When misalignments are found to be real (or ‘actual’) analysts decide how to respond to them by next considering whether responding to the misalignment is within the software scope or outside the software scope. Furthermore, the SPSVs studied both had a general aim of minimizing the level of customisation they engaged in. Analysts also spoke of software demonstrations in general as an opportunity for convincing users that there were alternative solutions to misalignments that had been identified. Analysts observed in my ethnographic study often attempted to show clients that the software on offer could actually meet their needs, even without customisation – it simply met them in an unexpected way.

8. Another area in which implementation analysts showed a desire to limit customisation appeared in relation to the issue of ‘work-arounds’. In packaged software, analysts use work-arounds in order to minimize customisation, rather than to reduce conflicts between requirements. The pronounced preference for minimal customisation seemed to be related to a) customisation’s association with increased costs and longer implementation time, and/or b) if the modifications requested by a client were extensive, this might have a significant impact on the software functions. The analysts voiced a general recommendation that clients should instead adopt the processes built into their software. In cases of ‘actual’ misalignments, sometimes analysts are left with no choice but to agree to customisation. However, in cases where a misalignment appeared to be actual and customisation was desired by the client but was outside the original scope of the software offer, the software company would provide the client with a report estimating the cost of a customisation. The cost was estimated by considering the extent of the misalignments and the customisation risk involved. In most cases, after receiving such a report, the clients chose to use the software as it was, and adapt their business process instead. Analysts did note, however, that if an organisation could afford a particular customisation, that customisation would likely go ahead.
9. There are potential benefits to SPSVs if misalignments are found. SPSVs may benefit from the identification of a misalignment because they can either improve their existing package immediately in response, or may be given a useful idea for the future enhancement of their software package in a future release.
10. There are three new practices connected to ‘Requirements Documents Practices’: creating a users’ needs/misalignments specification document, estimating the time and cost related to creating the users’ needs/misalignments document, and the use of a users’ needs validation document. Amongst other new findings, new requirements elicitation practices for PSIRE were also found, including using a live software demonstration to elicit the users’ needs and

making use of a user manual. Some of the elicitation practices in PSIRE that were found included using software demonstrations, creating live scenarios for testing packages, and prototyping poorly understood requirements. The way in which analysts were expected to prioritise requirements in PSIRE was also considered. It was discovered that analysts working with packaged software collect requirements in a circular process and develop those requirements that are agreed upon at the time (with clients) or that their managers award priority to. In PSIRE, therefore, prioritising requirements is a practice used at the discretion of project managers.

11. It is considered important in PSIRE for an analyst to be able to specify the relationship between users' needs and other software functions – that is, to know what functions are redundant in a specific use context for software and what software functions are essential. This is very important in PSIRE because analysts may be told to remove or modify package functions that would have a major effect on other functions of the software. A further practice that takes on additional importance in PSIRE is the need to document the links between stakeholder requirements and system models. This practice involves ensuring there is a high degree of traceability and rationale for a requirement. When it comes to requirements validation, I found that validation checklists were not used as commonly in PSIRE. Requirements in PSIRE were often instead validated by using prototyping to animate requirements, using a draft user manual, or by proposing requirements test cases.
12. A Parallel Star Model is proposed which details the processes involved in PSIRE. The star-shaped configuration of the Parallel Star Model shows the possible interconnections between different processes involved in the implementation of packaged software, and shows how, after analysts follow (part of) a particular process, they travel through a central step of 'assessment' before they make a decision about which action or activity to engage in next. In terms of PSIRE, the decisions that analysts make during these times of 'assessment' will relate to misalignments such as actual and perceived

misalignments, the preference to minimize customization, the benefits of misalignments, software scope, software price, client's organization size, new features risk, customization risk, output/input customization risk, technical needs, client's organization structure, and analysts' work domain.

13. The Parallel Star Model for PSIRE supports the use of interdependent, interwoven, processes. The Parallel Star Model shows not only that the processes involved in PSIRE are interconnected in flexible ways, but that multiple PSIRE processes can be followed simultaneously and in any order.

8.3. Contributions

Previous studies of packaged software implementation have typically focused on the users' perspective of implementation; this perspective revolves around the users' attempts to select packaged software that fits their process. However, the novel contribution of my study is that it changes the perspective used to examine packaged software implementation from 'outside' the SPSVs to 'inside' the small packaged software vendors (in Jordan, in this study) and therefore provides a new understanding of packaged software implementation and analysts' practices. This thesis reports an ethnographic study of packaged software implementation that also offers detailed discussion and analysis of the day-to-day practices of packaged software analysts. Through the research I carried out during this study I was able to successfully answer the main research question, which was: "What are the analysts' practices in the context of packaged software implementation by small packaged software vendors in Jordan?" I was able to gain a strong understanding of RE practice for PS implementation at SPSVs and of the various factors that bear influence upon it.

The first outcome of this thesis is that it provides an improved understanding of analysts' practices in terms of packaged software implementation requirements engineering, particularly as many of the major factors impacting on SPSVs' analysts' practices are discussed, from collecting initial requirements to identifying misalignments, to factors that impact on customisation decisions, to the use of software demonstrations. It is therefore considered that this PhD delivers an approach that could inform other SPSVs in relation to their conducting or managing a PS implementation in terms of RE practices, as it thoroughly explains issues such as analysts' roles during pre-implementation and implementation, the importance of various forms of software demonstration, mechanisms of scoping, and creating a packaged software offer, noting that the analyst conducting packaged software RE is often required to adopt an analyst-sales-marketing role. The following points express analysts' practices in terms of higher-level theoretical concepts:

- 'Live scenario software demonstration' and 'Knowledge and experience in packaged functions' were conceptualized as strategies intended to help analysts convince clients about a software solution, as they demonstrate and discuss a possible solution.
- 'Client's organisation structure and the limitations of analysts' company work domain' were conceptualized as assessment criteria that were intended to help analysts to make decisions about which software solution to demonstrate.
- Analysts' roles have changed from only collecting requirements and software analysis in traditional RE to also engaging in the pre-sale of software.
- The New Features request, Customisation needs, Software Output/Input, and Technical needs were conceptualized as assessment criteria that were intended to help the company to create a packaged software offer.
- Installing a copy of the software and software demonstration were conceptualized as strategies intended to help analysts identify misalignments (new features, transactions, output/input and technical).
- Actual misalignments, perceived misalignments, minimising customisation, software scope, organisation size, price of software, and misalignments benefits were conceptualized as assessment criteria that were intended to help analysts to make decisions about modification requests.

- Software demonstration was conceptualized as a strategy intended to help analysts to explain software functions, train users, and validate the modifications made.

The second outcome of this thesis is an in depth assessment of RE practices in terms of whether they were used in PSIRE, and, if used, to what degree they were used. Splitting the whole process of implementation into specific elements and activities, according to the use of particular documents or particular groupings of processes for collecting and validating requirements, has enabled a fine-grained view of activities and practices to be developed. In general terms, it was found that PSIRE introduced new practices of documentation, was not as concerned as general RE with looking for domain constraints or with collecting requirements and viewpoints from multiple sources, was more likely to involve live software demonstrations and screenshots to collect users' needs, and was more likely to involve the compilation of a user manual. In PSIRE, prioritising requirements is not a basic practice; instead, analysts collect requirements in a circular process, with managers then directing analysts regarding which requirements to give most attention toward. PSIRE was also found to place emphasis on assessing requirements risks and on considering the relationship between users' needs and the inter-relationships between software functions, as analysts engaging in PSIRE do not wish to disrupt functions of their software when making modifications in response to client requests. The assessment of PSIRE led to the discovery of new RE practices that were being used in four of the areas examined: new RE practices were found in relation to requirements documentation, requirements elicitation, requirements analysis and negotiation, and describing requirements.

The third and major outcome of this thesis is that it delivered an understanding of PSIRE that represents how SPSVs might identify and respond to misalignments between users' needs and PS functionalities. It does so through proposing a Parallel Star Model for PSIRE which is based on empirical observations made during my ethnographic research. The model is designed to depict the parallel nature of the processes conducted during PSIRE: feasibility study, assessment, implementation, software demonstration, and identifying misalignments. I have noted that analysts conducting PSIRE are commonly able to carry out multiple processes at the same time. Therefore, I hypothesise that PSIRE is actually conducted in terms of a parallel star process model rather than a linear process model. Further, the Parallel Star Model is appropriate in that these processes can happen at the same time *and in no particular order*.

8.4. Limitations

While concluding that I was ultimately successful with the results of my research, being able to answer my research questions and to generate new theory, I also acknowledge that there are some limitations to this study and my research results. While there were some specific limitations that appeared during the course of my study, it should be noted that the ethnographic study approach always brings its own limitations.

One particular limitation of ethnography is that, to some degree, interpretation of the results depends on the ethnographer. The ethnographer may obtain both quantitative and qualitative data, but after this, it is the ethnographer who chooses how to interpret the data and how to arrange narratives and explanations in particular ways. It is possible that a different ethnographer might have viewed the events observed and the SPSVs settings involved in a different way, or might have analysed the data collected in a different way. The contribution made by this thesis, therefore, represents what is at least partly my own personal perspective of what happened in the two SPSVs observed in Jordan, and my personal perspective of the various practices that make up requirements engineering for packaged software implementation. An ethnographer cannot pretend that their results and interpretations are entirely objective. My aim throughout the discussion of my ethnographic work as provided in this thesis was therefore to provide enough detailed information about the settings encountered, the information I was provided by analysts, the activities I witnessed analysts engaged in, the challenges that analysts told me about, and the data obtained during validation, that readers will be able to make their own firm judgments about the credibility of my study. If the ethnographer follows an ethical approach to data collection – an approach I attempted to maintain throughout my work – this also aids the credibility of ethnographic research.

Apart from these more general limitations that attend any ethnographic study, there were some more specific limitations that emerged as I conducted my ethnographic study of the SPSVs in Jordan. First, during the field work I conducted, there were some factors or processes involved in PSIRE that I was not able to investigate fully. For example, my work within the SPSVs revealed to me that writing a report helped analysts to evaluate the effort of time and cost that would be involved in customising a software package. However, the analysts chose not to share with me any specific information related to writing a report, so I was not able to fully comprehend how this kind of report influenced the PSIRE. Meanwhile, I was told that in the case where clients requested customisations that would make the software package implementation go outside the scope of the original software offer, users usually made the final decision of adapting their business process to match the software as it was, rather than follow through with the customisation request. I was unable to obtain any further details that would make it clear whether or why clients made such decisions, but it did appear from what analysts stated that the clients had probably been dissuaded by the increased time and cost issues (which would have been mentioned on reports they were furnished with).

One final limitation with regard to my ethnographic study is that on occasion, my study and its findings were constrained by needing to preserve issues of confidentiality.

8.5. Ideas for Future Research

As I suggested in my 'Discussion' chapter (Chapter 7), it could be desirable for researchers to shift their focus from examining issues from users' perspective to examining the views SPSVs have of packaged software implementation. If such action is taken, researchers and practitioners will be able to gain a more complete understanding of all of the sites and participants involved with packaged software implementation. Meanwhile, the results that I have gained from this study could be validated further if researchers gained more data about PSIRE from other SPSVs.

Another topic related to PSI that would be interesting to investigate is how the philosophy behind release planning for packaged software differs between large packaged software development companies and companies that are SPSVs. From observations and from previous literature, it appears, for example, that large packaged software development companies tend to have very detailed release plans and schedules for future packaged software products, planned out months or years in advance, while SPSVs may take a more ad hoc approach that instead involves continuous improvement of their product in response to clients' requirements and how clients feel about the product. Lastly, a further research area of interest would be an investigation of tools that could support misalignment management for SPSVs. If existing tools do not support the management of modifications to already developed packaged software functionalities other researchers might look to develop these in future work.

References

Achanga, P., Shehab, E., Roy, R., & Nelder, G. (2006). Critical success factors for lean implementation within SMEs. *Journal of Manufacturing Technology Management, 17*(4), 460-471.

Addo-Tenkorang, R., & Helo, P. (2011). *Enterprise Resource Planning (ERP): A Review Literature Report.* Paper presented at the the World Congress on Engineering and Computer Science, San Francisco, USA.

Ager, M (1980). The Professional Stranger: An Informal Introduction to Ethnographic, New York, Academic Pres

Ahmed, F. (2012). Software Requirements Engineer: An Empirical Study about Non-Technical Skills. *JOURNAL OF SOFTWARE, 7*(2).

Ahmed, F., Capretz, L. F., Bouktif, S., & Campbell, P. (2012). Soft skills requirements in software development jobs: a cross-cultural empirical study. *Journal of Systems and Information Technology, 14*(1), 58-81.

Akkermans, H., & Helden, K. v. (2002). Vicious and virtuous cycles in ERP implementation: a case study of interrelations between critical success factors. *European Journal of Information Systems, 11.*

Al-Ani, B., & Sim, S. E. (2006). *So, you think you are a requirements engineer?* Paper presented at the Requirements Engineering, 14th IEEE International Conference.

Al-Ani, B., & Sim, S. E. (2006). *Using expertise as a framework for evaluating requirements technology.* Paper presented at the Comparative Evaluation in Requirements Engineering, 2006. CERE'06. Fourth International Workshop on.

Allan, G. (2003). A critique of using grounded theory as a research method. *Electronic Journal of Business Research Methods, 2*(1), 1-10.

Al-Mashari, M., & Al-Mudimigh, A. (2003). ERP implementation: lessons from a case study. *Information Technology & People, 16*(1).

Alvarez, R. (2002). Confessions of an information worker: a critical analysis of information requirements discourse. *Information and Organization, 12*(2), 85-107.

Aranda, J. (2010). *Playing to the strengths of small organizations.* Paper presented at the Proceedings of the 1st Workshop on RE in Small Companies (RESC),[4].

Aranda, J., Easterbrook, S., & Wilson, G. (2007). *Requirements in the wild: How small companies do it.* Paper presented at the Requirements Engineering Conference, 2007. RE'07. 15th IEEE International.

Barney, S., Aurum, A., & Wohlin, C. (2008). A product management challenge: Creating software product value through requirements selection. *Journal of Systems Architecture, 54*(6), 576-593.

Beecham, S., Hall, T., & Rainer, A. (2003). Software process improvement problems in twelve software companies: An empirical analysis. *Empirical software engineering, 8*(1), 7-42.

Bernroider, E., & Koch, S. (2001). ERP selection process in midsize and large organizations. *Business Process Management Journal, 7*(3), 251-257.

Bingi, P., Sharma, M. K., & Godla, J. K. (1999). Critical issues affecting an ERP implementation. *Information systems management, 16*(3), 7-14.

Boyatzis, R. E. (1998). *Transforming qualitative information: Thematic analysis and code development*: SAGE Publications, Incorporated.

Browne, G. J., & Rameshb, V. (2002). Improving information requirements determination: a cognitive perspective. *Information and management, 39*(8), 625.

Browne, G. J., & Rogich, M. B. (2000). An Empirical Investigation of User Requirements Elicitation: Comparing the Effectiveness of Prompting Techniques. *Joumat of Management Information Systems, 17*(4), 223-249.

Buonanno, G., Faverio, P., Pigni, F., Ravarini, A., Sciuto, D., & Tagliavini, M. (2005). Factors affecting ERP system adoption: A comparative analysis between SMEs and large companies. *Journal of Enterprise Information Management, 18*(4), 384-426.

Bürsner, S., & Merten, T. (2010). RESC 2010: 1st Workshop on Requirements Engineering in Small Companies. *econstor*, 128.

Bürsner, S., Merten, T., Jantunen, S., Aranda, J., Saliou, P., & Ribaud, V. (2010). 6 First Workshop on

Requirements Engineering in Small Companies (RESC). *econstor*, 127.

Carnap, R. (1953). Testability and meaning (pp. pp-47). Appleton-Century-Crofts.

Charmaz, K. (2003). Grounded theory. *Strategies of qualitative inquiry, 2*, 249.

Chatzoglou, P. D. (1997). Factors affecting completion of the requirements capture stage of projects with different characteristics. *Information and Software Technology, 39*(9), 627-640.

Chau, P. Y. K. (1995). Factors used in the selection of packaged software in small businesses: Views of owners and managers. *Information & Management, 29*.

Chiasson, M., & Green, L. (2007). Questioning the IT artefact: user practices that can, could, and cannot be supported in packaged-software designs. *European Journal of Information Systems, 16*.

Chien, S.-W., Hu, C., Reimers, K., & Lin, J.-S. (2007). The influence of centrifugal and centripetal forces on ERP project success in small and medium-sized enterprises in China and Taiwan. *International Journal of Production Economics, 107*(2), 380-396.

Collis, J., & Hussey, R. (2009). *Business research: A practical guide for undergraduate and postgraduate students*: Palgrave Macmillan.

Corbin, J., & Strauss, A. (2007). *Basics of qualitative research: Techniques and procedures for developing grounded theory*: Sage Publications, Incorporated.

Coughlan, J., Lycett, M., & Macredie, R. D. (2003). Communication issues in requirements elicitation: a content analysis of stakeholder experiences. [Journal article (JA)]. *ScienceDirect, 45*(8), 525-537. doi: 10.1016/SO950-5849(03)00032-6

Cox, K., Niazi, M., & Verner, J. (2009). Empirical study of Sommerville and Sawyer's requirements engineering practices. *Software, IET, 3*(5), 339-355.

Damsgaard, J., & Karlsbjerg, J. (2010). Seven Principles for Selecting Software Packages. *Communications of the ACM, 58*(8).

Daneva, M. (2004). ERP Requirements Engineering Practice: Lessons Learned. *IEEE SOFTWARE*.

Daneva, M. (2007). *Understanding Success and Failure Profiles of ERP Requirements Engineering: an Empirical Study*. Paper presented at the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications.

Daneva, M., & Ahituv, N. (2012). *What agile ERP consultants think of requirements engineering for inter-organizational ERP Systems: Insights from a Focus Group in BeNeLux.* Paper presented at the Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on.

Daneva, M., & Wieringa, R. J. (2006). A requirements engineering framework for cross-organizational ERP systems. *Requirements Eng, 11*.

Davenport, T. H. (1998). Putting the enterprise into the enterprise system. *Harvard business review, 76*(4).

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., . . . Sitaram, P. (1993). *Identifying and measuring quality in a software requirements specification.* Paper presented at the Software Metrics Symposium, 1993. Proceedings., First International.

Denzin, N. K. (2000). Interpretive ethnography. *Zeitschrift für Erziehungswissenschaft, 3*(3), 401-409.

Denzin, N. K., & Lincoln, Y. S. (2005). *The Sage handbook of qualitative research*: Sage Publications, Incorporated.

Dittrich, Y., Vaucouleur, S., & Giff, S. (2009). ERP customization as software engineering: knowledge sharing and cooperation. *Software, IEEE, 26*(6), 41-47.

Easterbrook, S., Singer, J., Storey, M.-A., & Damian, D. (2008). Selecting empirical methods for software engineering research *Guide to advanced empirical software engineering* (pp. 285-311): Springer.

Eisenhardt, K. M. (1989). Building theories from case study research. *AcadManage Rev, 14*(4), 532–550.

El Emam, K., & Madhavji, N. H. (1995). *A field study of requirements engineering practices in information systems development.* Paper presented at the Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on.

Fossey, E., Epstein, M., Findlay, R., Plant, G., & Harvey, C. (2002). Creating a positive experience of research for people with psychiatric disabilities by sharing feedback. *Psychiatric rehabilitation journal, 25*(4), 369.

Gasser, L. (1986). The integration of computing and routine work. *ACM Transactions on Information Systems (TOIS), 4*(3), 205-225.

Ghobadian, A., & Gallear, D. (1996). Total quality management in SMEs. *Omega, 24*(1), 83-106.

Ghobadian, A., & Gallear, D. (1997). TQM and organization size. *International Journal of Operations & Production Management, 17*(2), 121-163.

Glaser, B. G. (1978). *Theoretical sensitivity: Advances in the methodology of grounded theory* (Vol. 2): Sociology Press Mill Valley, CA.

Gorschek, T., Gomes, A., Pettersson, A., & Torkar, R. (2012). Introduction of a process maturity model for market-driven product management and requirements engineering. *Journal of Software: Evolution and Process, 24*(1), 83-113.

Gregor, S. (2006). The nature of theory in information systems. *Mis Quarterly, 30*(3), 611-642.

Gregor, S. (2007). Design theory in information systems. *Australasian Journal of Information Systems, 10*(1).

Haddara, M., & Zach, O. (2011). *ERP systems in SMEs: A literature review.* Paper presented at the System Sciences (HICSS), 2011 44th Hawaii International Conference on.

Haines, M. N. (2009). Understanding enterprise system customization: An exploration of implementation realities and the key influence factors. *Information Systems Management, 26*(2), 182-198.

Hall, T., Beecham, S., & Rainer, A. (2002). Requirements problems in twelve software companies: an empirical analysis. *IEE Proceedings-Software, 149*(5), 153-160.

Hammersley, M., & Atkinson, P. (1995). Ethnography: principles in practice.

Hammersley, M., & Atkinson, P. (2007). Ethnography: principles in practice.

Hardiman, S. (2002). *REDEST-14 best practice SME experiments with innovative requirements gathering techniques.* Paper presented at the Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on.

Hartson, H. R., & Hix, D. (1989). Human-computer interface development: concepts and systems for its management. *ACM Computing Surveys (CSUR), 21*(1), 5-92.

Harvey, L. J., & Myers, M. D. (1995). Scholarship and practice: the contribution of ethnographic research methods to bridging the gap. *Information Technology & People, 8*(3), 13-27.

Holland, C. P., & Light, B. (1999). A Critical Success Factors Model For ERP Implementation. *IEEE Software*.

Holloway, I., & Wheeler, S. (2009). *Qualitative research in nursing and healthcare*: Wiley-Blackwell.

Holtzblatt, K., & Beyer, H. (1993). Making customer-centered design work for teams. *Communications of the ACM, 36*(10), 92-103.

Hong, K.-K., & Kim, Y.-G. (2002). The critical success factors for ERP implementation: an organizational fit perspective. *Information & Management, 40*(1), 25-40.

Huin, S. (2004). Managing deployment of ERP systems in SMEs using multi-agents. *International Journal of Project Management, 22*(6), 511-517.

Jantunen, S. (2010). *The benefit of being small: Exploring market-driven requirements engineering practices in five organizations.* Paper presented at the Proceedings of the 1st Workshop on RE in Small Companies (RESC),[4].

Jebreen, I., & Wellington, R. (2013). *Understanding Requirements Engineering Practices for Packaged Software Implementation.* Paper presented at the 4 th IEEE International Conference on Software Engineering and Service Sciences, Beijing, China.

Jebreen, I., Wellington, R., & MacDonell, S. (2013, September 2-4). *Understanding Feasibility Study Approach for Packaged Software Implementation by SMEs.* Paper presented at the 22nd International Conference on Information System Development, Seville, Spain.

Jebreen, I., Wellington, R., & MacDonell, S. G. (2013, 2-5 December). *Packaged Software Implementation Requirements Engineering by Small Software Enterprises.* Paper presented at the The 20th Asia-Pacific Software Engineering Conference (APSEC 2013), Bangkok, Thailand.

Jiang, L., & Eberlein, A. (2003). *Requirements engineering: a review and a proposal.* Paper presented at the Proceedings of the Third ASERC Workshop on Quantitative and Software Engineering.

Johansson, B., & Bjørn-Andersen, N. (2007). *Identifying Requirements for Future ERP Systems.* Paper presented at the Proceedings of the 30th Information Systems Research Seminar in Scandinavia IRIS.

Karlsson, L., Dahlstedt, Å., och Dag, J. N., Regnell, B., & Persson, A. (2003). *Challenges in market-driven requirements engineering-an industrial interview study.* Paper presented at the

Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02).

Karlsson, L., Dahlstedt, Å. G., Regnell, B., Natt och Dag, J., & Persson, A. (2007). Requirements engineering challenges in market-driven software development–An interview study with practitioners. *Information and Software technology, 49*(6), 588-604.

Kato, J., Saeki, M., Ohnishi, A., Nagata, M., Kaiya, H., Komiyaa, S., . . . Watahiki, K. (2003). *PAORE: package oriented requirements elicitation.* Paper presented at the Software Engineering Conference, 2003. Tenth Asia-Pacific.

Kendall, J. (1999). Axial coding and the grounded theory controversy. *Western Journal of Nursing Research, 21*(6), 743-757.

Khan, A. S. (2011). Software Requirement Engineering For Small and Medium Enterprise: A Case Study.

Khoo, H. M., & Robey, D. (2007). Deciding to upgrade packaged software: a comparative case study of motives, contingencies and dependencies. *European Journal of Information Systems, 16*.

Khoo, H. M., Robey, D., & Rao, S. V. (2011). An exploratory study of the impacts of upgrading packaged software: a stakeholder perspective. *Journal of Information Technology, 26*.

Kirsch, L. J., & Haney, M. H. (2006). Requirements determination for common systems: turning a global vision into a local reality. *Journal of Strategic Information Systems, 15*.

Kitto, S., & Higgins, V. (2010). Working around ERPs in technological universities. *Science, Technology & Human Values, 35*(1), 29-54.

Klein, H. K., & Myers, M. D. (2001). A classification scheme for interpretive research in information systems. *Qualitative Research in IS: Issues and Trends*, 218-239.

Krueger, R. A., & Casey, M. A. (2008). *Focus groups: A practical guide for applied research*: SAGE Publications, Incorporated.

Laplante, P. A., Neill, C. J., & Jacobs, C. (2002). *Software requirements practices: some real data.* Paper presented at the Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE.

Laukkanen, S., Sarpola, S., & Hallikainen, P. (2007). Enterprise size matters: objectives and constraints of ERP adoption. *Journal of Enterprise Information Management, 20*(3), 319-334.

Lee, A. S., & Baskerville, R. L. (2003). Generalizing generalizability in information systems research. *Information systems research, 14*(3), 221-243.

Lee, D. M., Trauth, E. M., & Farwell, D. (1995). Critical skills and knowledge requirements of IS professionals: a joint academic/industry investigation. *Mis Quarterly*, 313-340.

Leffingwell, D., & Widrig, D. (2003). *Managing software requirements: a use case approach*: Addison-Wesley Professional.

Lehtola, L., & Kauppinen, M. (2006). Suitability of requirements prioritization methods for market-driven software product development. *Software Process: Improvement and Practice, 11*(1), 7-19.

Light, B. (2005). Going beyond 'misfit' as a reason for ERP package customisation. *Computers in Industry, 56*.

Light, B., & Sawyer, S. (2007). Locating packaged software in information systems research. *European Journal of Information Systems, 16*.

Loh, T. C., & Koh*, S. (2004). Critical elements for a successful enterprise resource planning implementation in small-and medium-sized enterprises. *International journal of production research, 42*(17), 3433-3455.

Luo, W., & Strong, D. M. (2004). A framework for evaluating ERP implementation choices. *Engineering Management, IEEE Transactions on, 51*(3), 322-333.

Mabert, V. A., Soni, A., & Venkataramanan, M. (2003). Enterprise resource planning: managing the implementation process. *European Journal of Operational Research, 146*(2), 302-314.

Maiden, N. A., & Ncube, C. (1998). Acquiring COTS software selection requirements. *Software, IEEE, 15*(2), 46-56.

Malhotra, R., & Temponi, C. (2010). Critical decisions for ERP integration: Small business issues. *International Journal of Information Management, 30*(1), 28-37.

McAdam, R. (2000). Quality models in an SME context: a critical perspective using a grounded approach. *International Journal of Quality & Reliability Management, 17*(3), 305-323.

McAdam, R. (2002). Large Scale Innovation Reengineering Methodology in SMEs Positivistic and Phenomenological Approaches. *International Small Business Journal, 20*(1), 33-52.

Merten, T., Lauenroth, K., & Bürsner, S. (2011). Towards a new understanding of small and medium sized enterprises in requirements engineering research *Requirements Engineering: Foundation for Software Quality* (pp. 60-65): Springer.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*: Sage Publications, Incorporated.

Mishra, D., & Mishra, A. (2009). *Market-Driven Software Project through Agility: Requirements Engineering Perspective.* Paper presented at the Business Information Systems Workshops.

Mohamed, A., Ruhe, G., & Eberlein, A. (2007). MiHOS: an approach to support handling the mismatches between system requirements and COTS products. *Requirements Engineering, 12*(3), 127-143.

Moon, Y. (2007). Enterprise Resource Planning (ERP): a review of the literature. *Mechanical and Aerospace Engineering*.

Morabito, V., Pace, S., & Previtali, P. (2005). ERP marketing and Italian SMEs. *European Management Journal, 23*(5), 590-598.

Morgan, D. L. (1996). *Focus groups as qualitative research* (Vol. 16): SAGE Publications, Incorporated.

Muscatello, J. R., Small, M. H., & Chen, I. J. (2003). Implementing enterprise resource planning (ERP) systems in small and midsize manufacturing firms. *International Journal of Operations & Production Management, 23*(8), 850-871.

Myers, M. (1999). Investigating information systems with ethnographic research. *Communications of the AIS, 2*(4es), 1.

Myers, M. D., & Avison, D. E. (2002). *Qualitative Research in Information Systems*. London: Sage.

Ncube, C., & Maiden, N. A. (1999). *Guiding parallel requirements acquisition and COTS software selection.* Paper presented at the Requirements Engineering, 1999. Proceedings. IEEE International Symposium on.

Ncube, C., & Maiden, N. A. (1999). *PORE: Procurement-oriented requirements engineering method for the component-based systems engineering development paradigm.* Paper presented at the International Workshop on Component-Based Software Engineering.

Newman, M., & Zhao, Y. (2008). The process of enterprise resource planning implementation and business process re-engineering: tales from two Chinese small and medium-sized enterprises. *Information Systems Journal, 18*(4), 405-426.

Nikula, U., Sajaniemi, J., & Kälviäinen, H. (2000). *A State-of-the-practice Survey on Requirements Engineering in Small-and Medium-sized Enterprises*: Lappeenranta University of Technology.

Niu, N., Jin, M., & Cheng, J.-R. C. (2011). A case study of exploiting enterprise resource planning requirements. *Enterprise Information Systems, 5*(2), 183-206.

Noll, C. L., & Wilkins, M. (2002). Critical skills of IS professionals: A model for curriculum development. *Journal of Information Technology Education, 1*(3), 143-154.

Nordheim, S., & Päivärinta, T. (2004). *Customization of Enterprise Content Management Systems: An Exploratory Case Study.* Paper presented at the 37th Hawaii International Conference on System Sciences, Hawaii.

Olsen, K. A., & Sætre, P. (2007). ERP for SMEs–is proprietary software an alternative? *Business Process Management Journal, 13*(3), 379-389.

Olsen, K. A., & Sætre, P. (2007). IT for niche companies: is an ERP system the solution? *Information Systems Journal, 17*(1), 37-58.

Olson, D. L., & Staley, J. (2012). Case study of open-source enterprise resource planning implementation in a small business. *Enterprise Information Systems, 6*(1), 79-94.

Olsson, T., Doerr, J., Koenig, T., & Ehresmann, M. (2005). A flexible and pragmatic requirements engineering framework for SME. *Proceedings: Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes*, 1-12.

Orlikowski, W. J. (1993). CASE tools as organizational change: investigating incremental and radical changes in systems development. *Mis Quarterly, 17*(3), 309-340.

Penzenstadler, B., Haller, G., Schlosser, T., & Frenzel, G. (2009). *Soft Skills REquired: A practical approach for empowering soft skills in the engineering world.* Paper presented at the

Requirements: Communication, Understanding and Softskills, 2009 Collaboration and Intercultural Issues on.

Pino, F. J., García, F., & Piattini, M. (2008). Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Journal, 16*(2), 237-261.

Pitts, M., & Browne, G. (2007). Improving requirements elicitation: an empirical investigation of procedural prompts. *Information systems journal, 17*(1), 22. doi: 10.1111/j.1365-2575.2006.00240.x

Plant, R., & Willcocks, L. (2007). Critical Success Factors In International ERP Implementations: A Case Research Approach. *Journal of Computer Information Systems*.

Poba-Nzaou, P., & Raymond, L. (2013). Custom Development as an Alternative for ERP Adoption by SMEs: An Interpretive Case Study. *Information Systems Management, 30*(4), 319-335.

Pohl, K. (2010). *Requirements engineering: fundamentals, principles, and techniques*: Springer Publishing Company, Incorporated.

Pollock, N. (2005). When is a work-around? Conflict and negotiation in computer systems development. *Science, Technology & Human Values, 30*(4), 496-514.

Puschmann, T., & Alt, R. (2005). Developing an integration architecture for process portals. *European Journal of Information Systems, 14*.

Quiescenti, M., Bruccoleri, M., La Commare, U., La Diega, S. N., & Perrone, G. (2006). Business process-oriented design of Enterprise Resource Planning (ERP) systems for small and medium enterprises. *International Journal of Production Research, 44*(18-19), 3797-3811.

Quispe, A., Marques, M., Silvestre, L., Ochoa, S. F., & Robbes, R. (2010). *Requirements Engineering Practices in Very Small Software Enterprises: A Diagnostic Study.* Paper presented at the Chilean Computer Science Society (SCCC), 2010 XXIX International Conference of the.

Radatz, J., Geraci, A., & Katki, F. (1990). IEEE standard glossary of software engineering terminology. *IEEE Std, 610121990*, 121990.

Regnell, B., Beremark, P., & Eklundh, O. (1998). A market-driven requirements engineering process: results from an industrial process improvement programme. *Requirements Engineering, 3*(2), 121-129.

Regnell, B., Höst, M., Natt och Dag, J., Beremark, P., & Hjelm, T. (2000). *Visualization of agreement and satisfaction in distributed prioritization of market requirements.* Paper presented at the Proceedings of 6th International Workshop on Requirements Engineering: Foundation for Software Quality.

Regnell, B., Höst, M., och Dag, J. N., Beremark, P., & Hjelm, T. (2001). An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering, 6*(1), 51-62.

Robert, Y. (1994). Case Study Research: design and methods. *LA: Sage Publication Inc*.

Robey, D., Ross, J. W., & Boudreau, M.-C. (2002). Learning to implement enterprise systems: an exploratory study of the dialectics of change. *Journal of Management Information Systems, 19*(1), 17-46.

Robey, D., Ross, J. W., & Boudreau, M.-C. (2002). Learning to implement enterprise systems: an exploratory study of the dialectics of change. *Journal of Management Information Systems, 19*(1), 17-46.

Robinson, A. L., Donahue, N. M., Shrivastava, M. K., Weitkamp, E. A., Sage, A. M., Grieshop, A. P., ... & Pandis, S. N. (2007). Rethinking organic aerosols: Semivolatile emissions and photochemical aging. Science, 315(5816), 1259-1262.

Rolland, C., & Prakash, N. (2000). Bridging the gap between organisational needs and ERP functionality. *Requirements Engineering, 5*(3), 180-193.

Rothenberger, M. A., & Srite, M. (2009). An investigation of customization in ERP system implementations. *Engineering Management, IEEE Transactions on, 56*(4), 663-676.

Rouibah, K., & Al-Rafee, S. (2009). Requirement engineering elicitation methods. A Kuwaiti empirical study about familiarity, usage and perceived value. *Information management & computer security, 17*(3), 192 - 217. doi: 10.1108/09685220910978086

Rowland, N. J. (2012). Prioritizing Packaged Software Implementation Projects: The Significance of

Gaps. *Phenomenology, Organizational Politics, and It Design: The Social Study of Information Systems*, 159.

Rowland, N. J., & Gieryn, T. F. (2008). 12 Transfer Troubles: Outsourcing Information Technology in Higher Education. *Living in a Material World*, 375.

Saiediana, H., & Dale, R. (2000). Requirements engineering: making the connection between the software developer and customer. *Information and Software Technology, 42*.

Saliou, P., & Ribaud, V. (2010). *Iso-standardized requirements activities for very small entities.* Paper presented at the Requirements Engineering in Small Companies 2010, 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2010).

Sandelowski, M. (2000). Focus on Research Methods-Whatever Happened to Qualitative Description? *Research in nursing and health, 23*(4), 334-340.

Sawyer, S. (2000). Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems, 9*.

Sawyer, S. (2001). A Market-Based Perspective on Information Systems Development. *Communications of the ACM, 44*(11).

Schultze, U. (2000). A confessional account of an ethnography about knowledge work. *MIS quarterly*, 3-41.

Scott, J. E., & Vessey, I. (2002). Managing Risks in Enterprise Systems Implementations. *Communications of the ACM, 45*(4).

Sharif, A. M., Irani, Z., & Love, P. E. (2005). Integrating ERP using EAI: a model for post hoc evaluation. *European Journal of Information Systems, 14*.

Shaul, L., & Tauber, D. (2012). CSFs along ERP life-cycle in SMEs: a field study. *Industrial Management & Data Systems, 112*(3).

Shih-Chieh, J., Lin, T.-C., Zheng, G.-T., & Hung, Y.-W. (2012). Users as knowledge co-producers in the information system development project. *International Journal of Project Management, 30*.

Sia, S. K., & Soh, C. (2007). An assessment of package–organisation misalignment: institutional and ontological structures. *European Journal of Information Systems, 16*.

Sims, C. A. (1986). Are forecasting models usable for policy analysis? *Federal Reserve Bank of Minneapolis Quarterly Review, 10*(1), 2-16.

Sims, C. A. (1998). Comment on Glenn Rudebusch's" Do measures of monetary policy in a VAR make sense?". *International Economic Review, 39*(4), 933-941.

Snider, B., da Silveira, G. J., & Balakrishnan, J. (2009). ERP implementation at SMEs: analysis of five Canadian cases. *International Journal of Operations & Production Management, 29*(1), 4-29.

Soh, C., Kien Sia, S., Fong Boh, W., & Tang, M. (2003). Misalignments in ERP implementation: a dialectic perspective. *International Journal of Human-Computer Interaction, 16*(1), 81-100.

Soh, C., Kien, S. S., & Tay-Yap, J. (2000). Enterprise resource planning: cultural fits and misfits: is ERP a universal solution? *Communications of the ACM, 43*(4), 47-51.

Soja, P. (2006). Success factors in ERP systems implementations: lessons from practice. *Journal of Enterprise Information Management, 19*(6).

Solemon, B., Sahibuddin, S., & Ghani, A. A. A. (2009). Requirements engineering problems and practices in software companies: An industrial survey *Advances in Software Engineering* (pp. 70-77): Springer.

Solemon, B., Sahibuddin, S., & Ghani, A. A. A. (2012). A New Maturity Model for Requirements Engineering Process: An Overview. *Journal of Software Engineering and Applications, 5*(5), 340-350.

Somers, T. M., & Nelson, K. (2001). *The Impact of Critical Success Factors across the Stages of Enterprise Resource Planning Implementations.* Paper presented at the the 34th Hawaii International Conference on System Sciences.

Sommerville, I. (2004). *Software engineering* (Seven Edition ed.): Addison-Wesley Publishers Limited.

Sommerville, I., & Kotonya, G. (1998). *Requirements engineering: processes and techniques*: John Wiley & Sons, Inc.

Sommerville, I., & Sawyer, P. (1997). *Requirements engineering: a good practice guide*: John Wiley &

Sons, Inc.

Spradley, J. P. (1979). The ethnographic interview.

Staehr, L., Shanks, G., & Seddon, P. B. (2012). An Explanatory Framework for Achieving Business Benefits from ERP Systems. *Journal of the Association for Information Systems, 13*(6), 2.

Stewart, D. W., Shamdasani, P. N., & Rook, D. W. (2006). *Focus groups: Theory and practice* (Vol. 20): SAGE Publications, Incorporated.

Strauss, A., & Corbin, J. (1994). Grounded theory methodology. *Handbook of qualitative research*, 273-285.

Strauss, A., & Corbin, J. (1998). Basics of qualitative research: Procedures and techniques for developing grounded theory: Thousand Oaks, CA: Sage.

Strauss, A., & Corbin, J. (1998). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*: Sage Publications, Inc.

Thomas, D. R. (2006). A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation, 27*(2), 237-246.

Trauth, E. M. (2001). *Qualitative research in IS: issues and trends*: IGI Global.

Umble, E. J., Haft, R. R., & Umble, M. M. (2003). Enterprise resource planning: Implementation procedures and critical success factors. *European Journal of Operational Research, 146*.

Urquhart, C. (2001). An encounter with grounded theory: tackling the practical and philosophical issues. *Qualitative Research in IS: Issues and Trends*, 104-140.

van Beijsterveld, J. (2006). *Misfits in ERP System Implementation.* Master thesis

Van Maanen, J. (2006). Ethnography then and now. *Qualitative Research in Organizations and Management: An International Journal, 1*(1), 13-21.

Van Maanen, J. (2011). Ethnography as work: some rules of engagement. *Journal of management studies, 48*(1), 218-234.

Van Manen, M. (1990). *Researching lived experience: Human science for an action sensitive pedagogy*: Suny Press.

Vilpola, I., Kouri, I., & Vaananen-Vainio-Mattila, K. (2007). *Rescuing Small and Medium-Sized Enterprises from Inefficient Information Systems--A Multi-disciplinary Method for ERP System Requirements Engineering.* Paper presented at the System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on.

Volkoff, O., Strong, D. M., & Elmes, M. B. (2005). Understanding enterprise systems-enabled integration. *European Journal of Information Systems, 14*.

Volkoff, O., Strong, D. M., & Elmes, M. B. (2007). Technological embeddedness and organizational change. *Organization Science, 18*(5), 832-848.

Wagner, E. L., Newell, S., & Piccoli, G. (2010). Understanding Project Survival in an ES Environment: A Sociomaterial Practice Perspective. *Journal of the Association for Information Systems, 11*(5).

Walsham, G. (2006). Doing interpretive research. *European Journal of Information Systems, 15*(3), 320-330.

Wei, H.-L., Wang, E. T. G., & Ju, P.-H. (2005). Understanding misalignment and cascading change of ERP implementation: a stage view of process analysis. *European Journal of Information Systems, 14*.

Whetten, D. A. (1989). What constitutes a theoretical contribution?. Academy of management review, 14(4), 490-495

Wiegers, K. (1999). First things first: prioritizing requirements. *Software Development, 7*(9), 11-19.

Wolcott, H. F. (1994). *Transforming qualitative data: Description, analysis, and interpretation*: Sage.

Wong, K. Y., & Aspinwall, E. (2004). Characterizing knowledge management in the small business environment. *Journal of Knowledge management, 8*(3), 44-61.

Xu, L., & Brinkkemper, S. (2005). *Concepts of Product Software: Paving the Road for Urgently Needed Research.* Paper presented at the Proceedings of the First International Workshop on Philosophical Foundations of Information Systems Engineering.

Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information Systems, 16*.

Yen, T. S., Idrus, R., & Yusof, U. (2011). A Framework for classifying misfits between enterprise resource planning (ERP) systems and business strategies. *Asian Academy of Management*

Journal, 16(2), 53-75.

Yeow, A., & Sia, S. K. (2008). Negotiating ‘‘best practices’’ in package software implementation. *Information and Organization, 18*.

Yin, R. K. (2008). *Case study research: Design and methods* (Vol. 5): SAGE Publications, Incorporated.

Zach, O., & Munkvold, B. E. (2012). Identifying reasons for ERP system customization in SMEs: a multiple case study. *Journal of Enterprise Information Management, 25*(5), 462-478.

Appendices

Appendix A: Participant information sheet

Participant Information Sheet

Date Information Sheet Produced: 07 February 2011

Project Title

Communication Strategy in Requirement Determination: an Exploration of Developers-users Interaction/ Nonverbal Channels Perspective

An Invitation

My name is Issam Jebreen. And I am currently doing a PHD at AUT University, Please accept my invitation to participate in this research project. In this research I am going to be investigating the requirement engineering process.

Your participation is voluntary and without any financial remuneration. Your contribution will be vital and important to the outcome. You will get the chance to get to know new communication skills and also increase the effectiveness of your communication process.

You can withdraw your contribution at any time before the data analysis is completed and if you decided to do so, there will be no disadvantage for you.

What is the purpose of this research?

The purpose of the research is to explore the themes and strategies to reach a shared understanding of the requirements from communication perspective.

How was I identified and why am I being invited to participate in this research?

You have been identified as a developer with expertise in the software development process. Your name has been put forward through a recommendation from a colleague or peer in the industry

What will happen in this research?

Interview Protocol: First of all, the meeting will take place at a mutually agreed upon time and place, and should last about 60 minutes or for as long as you want to speak with me. I will use the video records from the Observation sessions conducted prior to the Interview. After, I will present and demonstrate open ended questions to you that I would like us to discuss. The questions will be based on the video records and will raise questions related to the research. The session will be audio recorded so that I can accurately reflect on what is discussed. The tapes will only be reviewed by members of the research team who will transcribe and analyze them. They will then be destroyed. If at any time and for any reason, you would prefer not to answer any questions, please feel free not to. If at any time you would like to stop participating, please tell me. We can take a break, stop and continue at a later date, or stop altogether. You will not be penalized in any way for deciding to stop participation at any time.

What are the discomforts and risks?

The only potential risk from your participation in this project is that there is a risk that confidential information about the user requirements determined for this project may become known to others in your organisation.

How will these discomforts and risks be alleviated?

In order to protect your privacy and safety, I will not use your name (unless you give me permission to do otherwise). In my notes, your name will be codified, and I may change your name for publication/presentation purposes. This information will be securely stored and the audiotape of this observation and interview will be destroyed once its contents are transcribed. The research data might be published approximately 1 year after it is collected and therefore any time dependant information will be passed by the time of publication.

You can have short breaks at any time for answering phone calls, going to the bathroom, or any other personal need.

What are the benefits?

You may benefit from actively participating in research and getting to know a topic that might be new and exciting for you. Your contribution is a part of my PHD Thesis,

Although you probably won't benefit directly from participating in this study, we hope that others in the community/society in general will benefit by the research results

How will my privacy be protected?

Your privacy will be protected at all times. Your personal data remains confidential. In order to achieve privacy and confidentiality, the observation and video tapes, as well as the interview and audio file will be identified only as a unique code which is not linked to any personal data. The video tapes will only be accessed by the researcher.

Study information will be kept in a secure location at the Auckland University. The results of the study may be published or presented at professional meetings, but your identity will not be revealed, which means that no one (not even the research team) will know what your answers are. So, please do not write your name or other identifying information on any of the study materials.

Also your name, address, or any other personal data will not be used for analysis or contribution to the research. However, your name and agreement to participate in this research will be disclosed to the manager of the organisation. If your manager wants to know who is taking part in my project I will check with you first.

What are the costs of participating in this research?

The only cost to you will be your time. The observation is expected to take about 90 minutes and the interview about 60 minutes.

What opportunity do I have to consider this invitation?

From the point of invitation to the point of acceptance of invitation, you will have at least 24 hours to consider this invitation.

How do I agree to participate in this research?

In order to participate, you need to read, understand, agree, and sign the attached consent form.

Will I receive feedback on the results of this research?

Feedback is available to every participant. The consent form contains an option box and space for contact details to express your interest in receiving feedback. The feedback will be sent to you after completion of the PHD thesis. The researcher expects this to happen in February 2013.

What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Project Supervisor, Dr. Judith Symonds, judith.symonds@aut.ac.nz, +64 9-921 9999 x5879

Concerns regarding the conduct of the research should be notified to the Executive Secretary, AUTEC, Madeline Banda, *madeline.banda@aut.ac.nz* , 921 9999 ext 8044.

Whom do I contact for further information about this research?

Researcher Contact Details:

Issam Jebreen, issam.jebreen@aut.ac.nz, +64 27 8595234, +64 9-921 9999 x5125

Project Supervisor Contact Details:

Dr Robert Wellington, robert.wellington@aut.ac.nz, +64 9-921 9999 x5879

Approved by the Auckland University of Technology Ethics Committee on 07/02/11, AUTEC Reference number 10/293.

Appendix B: Initial Codes by Key point coding and Statistical analysis

"Pre-implementation was one of the processes that analysts applied during the PSIRE. This process occurred as a standard initial stage and was used by analysts during the entire time when I undertook my observation process and interviews. It was a process that was also applied during any demonstration presentation situation"

The dominant codes involved in Pre-implementation are below.

Pre-implementation	ID	codes	Paraphrased from interpretive data
	I-D1	Explaining product functionality	Presenting the potential product to the client.
	I-D2	Identifying possible improvements	Asking the clients about specific features for the software Discussing possibilities to improve, add, or modify some functions for client's satisfaction.
	I-D3	Discovering client's initial needs	Identifying the client's expectations and the main objectives for using the software.
	I-D4	Linking business process and product functionality	Presenting the functionality of the software and identifying its need according to the business process.
	I-D5	Discussing client's issues	Identifying client's issues within his business process Tackling the issues in the software.
	I-D6	Prototyping presentation	Presenting the software through the means of screen shots in order to help the client understand and see how this product could solve and satisfy their business process issues.

In-depth Description

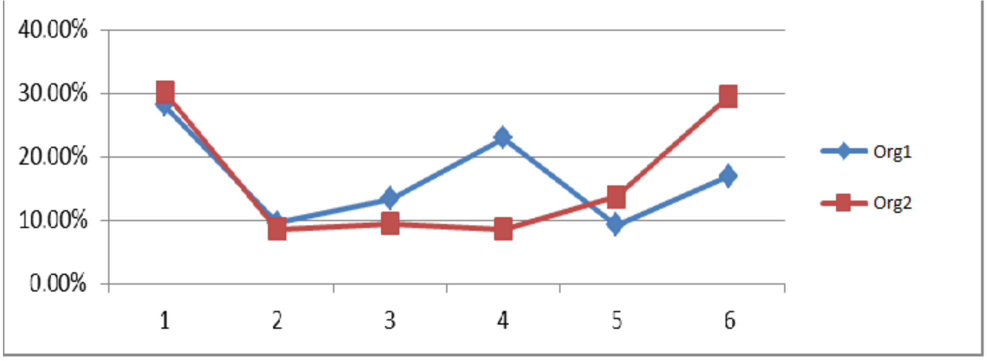
ID	Paraphrased from interpretive data
I-D1	Explaining the product's potential: analysts represented the product that the client might be interested in.
I-D2	Identifying possible improvements: clients asked about some features that they would like to see in the software. Analysts and clients discussed the possibilities to improve, add, or modify some functions in order to fit with the client's expectation.
I-D3	Discovering the client's initial needs: analysts tried to figure out the client's expectations of the software and the main objectives for using the software.
I-D4	Linking business process and product functionality: analysts represented the potential product functionality from the point of view of how it suited the business process, and explained why this functionality would be satisfactory.
I-D5	Discussing the client's issues: analysts tried to understand the challenges that faced the client within his business process and tried to represent how the potential product could deal with and tackle these challenging issues.
I-D6	Prototyping presentation: analysts used some screen shots to represent the potential product functionality in order to demonstrate to the client or convince the client how this product could satisfy their business process and solve any problematic issues in their business process.

Based on the percentage results of codes inducted from Org1 during the pre-implementation, I can observe that the most inducted code was explaining product functionality (28.1%), while the least inducted code was discussing client's issues (9.2%). The second most commonly inducted code linking business process and product functionality (22.9%), followed by prototyping presentation (16.9%) and discovering client's initial needs (13.3). The second least commonly inducted code was identifying possible improvements, with the percentile usage of 9.6%.

Percentage of codes inducted from organization

pre-implementation stage,	ID	Org1	ID	Org2
	I-D5	9.20%	I-D2	8.60%
	I-D2	9.60%	I-D4	8.60%
	I-D3	13.30%	I-D3	9.40%
	I-D6	16.90%	I-D5	13.70%
	I-D4	22.90%	I-D6	29.50%
	I-D1	28.10%	I-D1	30.20%

With Org2, the inducted code of explaining product functionality was inducted the most as well – at a rate of 30.2%. Only slightly behind this in terms of inducted code was prototyping presentation (29.5%). The inducted code of the discussion of client's issues was used at 13.7%, followed by discovering client's initial needs strategy (9.4%). The inducted codes of identifying possible improvements and linking business process and product functionality were range the least, with a result of 8.6% each.



"Marking and sales team at the pre-implementation stage make contact with the potential *client and to present the company's services and potential business relations. Phone calls, emails and personal visit have conducted to provide marketing information and potential product related to the client's business, including past examples of the products developed for a similar field of users, in order to show the competence of the analysts' organisation and to present the best situations to the potential client. This approach helps the potential client to learn about the software development organisation and compare the information from them with that from other similar organisations. It is vital at this stage that the marketing team presents the organisation in its best light".*

Pre-implementation

pre-implementation	ID	codes	Paraphrased from interpretive data
	I-D7	Marketing	Presenting the organisation's information Provisioning the proof of the organisation's competence in the clients' field
	I-D8	Meetings arrangement	Arranging the meetings Confirming the meetings
	I-D9	Documents exchange	Exchanging the documents suck company profile, contact, business cards.
	I-D10	Official documents exchange	Presenting the official documents for software product information

In-depth Description for inducted codes

ID	Paraphrased from interpretive data
I-D7	The marketing and sales team presented their organisation's information, which showed the

	clients the stability of the organisation, the time period the organisation had been in business or presented a particular service provided by their company, as well as company's credibility. The clients were also presented with information relating to the field of work the organisation dealt with, as the organisation provided examples of the previous work done for clients from the same field or a similar field. The developers also displayed confidence in their work. The emails also provided verbatim quotations from previous clients, showing their satisfaction with the work done and stating or demonstrating that it improved their business process, as the users' workload was eased after the new software was developed.
I-D8	The marketing and sales team try to arrange meetings for demonstration presentations, which were held at either the analysts' or clients' company meeting space. Meetings were arranged to be suitable with the clients' schedules. They advised on the meeting date, time, and place. Closer to the arranged meeting date, the marketing and sales team sent a confirmation email or call about the meeting arrangement, confirming with the client that it was still taking place.
I-D9	After the meeting was held, the analysts emailed meetings minutes to the client. The organisation's profile was presented as well, in an Email.
I-D10	The marketing and sales team provide the product information, company profile and company references

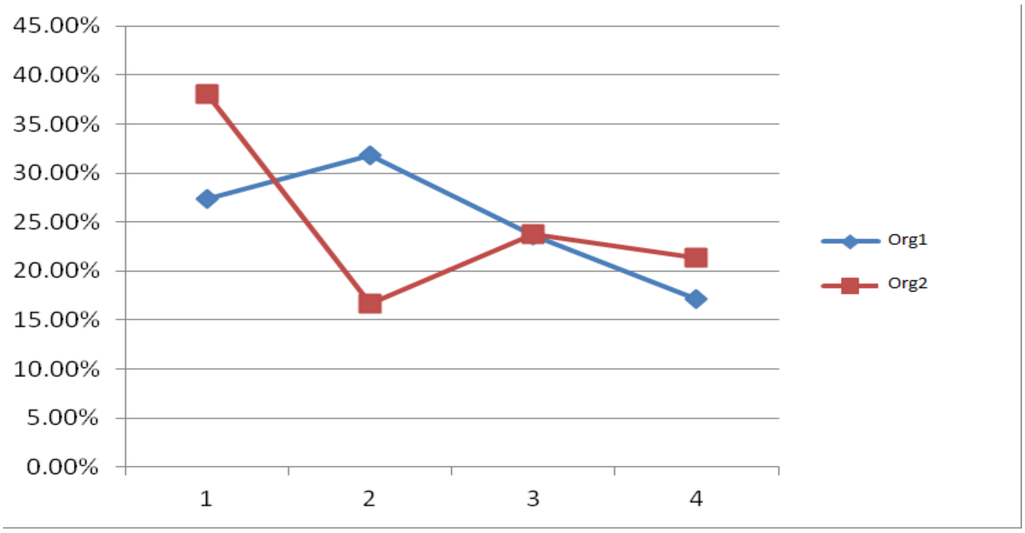
Marketing code was inducted from first organisation with 57%. The second organisation was inducted only 43% for the same code. There were 78% and 22% support for meetings arrangement by Organisation 1 and Organisation 2 companies, respectively. Organisation 1 inducted the document exchange code with 65%, while Organisation 2 inducted the same code by only 35%. There were 60% and 40% inducted for the official exchange document code in Organisation 1 and Organisation 2 organisations respectively.

Percentage of inducted codes by organization

pre-implementation stage	ID	Org1	ID	Org2
	I-D10	17.20%	I-D8	16.70%
	I-D9	23.60%	I-D10	21.40%
	I-D7	27.40%	I-D9	23.80%
	I-D8	31.80%	I-D7	38.10%

As shown by the percentile inducted codes of each Organisation 1 at pre-implementation stage, the most inducted code was meetings arrangement (31.8%), followed by marketing (27.4%) and documents exchange (23.6%) codes. The least inducted code was official documents exchange (17.2%). Organisation 2, on the other hand, marketing code was the most (38.1%) inducted, followed by documents exchange and official documents exchange codes (23.8% and 21.4%, respectively). The least inducted code was meetings arrangement, with 16.7%.

When the two companies were compared, the results showed that the documents exchange inducted code at about the same level, while official documents exchange was inducted code more by Organisation 2. While Organisation 2 inducted marketing code more than Organisation 1, the latter took more advantage of utilising the meetings arrangement code. The graph below visually represents the comparison of each inducted code by both companies at the pre-implementation stage,.



After the software offer is accepted by the client, we use the information from pre-implementation stage to start collecting more details about the user's needs… So first, you install a copy of our software then we start to explain our software functions to determine mismatches between our software and the user's business process. USER2- Formal Interview (08/02/2012)

We cannot ask users about their infrastructure because most of the users are not IT people and even IT people don't know some of the infrastructure requirements for software to be run……We try to discover any issues with the users' infrastructure, but we do that by installing a copy of our software … that's the only way to get to know the issues with the users' infrastructure…. USER2- Formal Interview (09/02/2012)

"Anyway, it is all about solving client issues and these needs more discussion and understanding to his/her business process are required" USER5- Formal Interview (23/03/2012)

"We use the initial report from demo panel and then we collect some information about client business to prepare for more understanding" USER1- Informal Interview

We start explaining the software functions. So we install a copy of the software, not to be used in a real process, but to help us to explain our software functions to users….. I think that helps the users know what software can do…. As you know, users sometimes have no understanding of what software can do, so we use a copy of our software to teach them, and to help users to engage in a good discussion about the software functions and their business process. USER1- Informal Interview

During Implementation inducted codes

During Implementation	ID	codes	Paraphrased from interpretive data
	In-D1	Build a Relation	Analysts built a relationship with the client in order to help the client express their thoughts in a confident way.
	In-D2	Explanation Tools	Analysts used different tools that could help explain the functionality of the software.
	In-D3	Body language	Analysts used their body language to send messages, and also interpreted body language in order to recognise the client's level of understanding.
	In-D4	In-depth Discussion	Analysts tried to get details of the client's business process and the requirements related to the product.
	In-D5	Requirement Validation	Analysts tried to validate the requirements once they collected further details about them.
	In-D6	Understand Database Structure	Analysts analysed client's data process within the requirements and the possible field in the database.

	In-D7	Identify Possible improvements	Analysts asked the clients about specific features needed in the software Analysts discussed possibilities of improving, adding, or modifying some functions for the client's satisfaction.
	In-D8	Discussion of client's issues	Analysts identified client's issues within their business process Analysts tackled the issues by changing the software.
	In-D9	Acquire knowledge	Analysts read, gathered and studied the information about the business process related to the client's organisation.
	In-D10	Clarify Information	Analysts clarified the gathered information with the client.
	In-D11	Define users roles, Key Users	Analysts obtained an understanding of each user's involvement in the software. Analysts obtained an understanding of how the requirements are related to each user. Analysts obtained an understanding of which functionalities of the software each user would have access to.
	In-D12	Explain business process	Clients and users explained the business activities and process.
	In-D13	Get confirmation of process	Analysts confirmed the business process with the client.
	In-D14	Open questions	Analysts invited the users to discuss their requirements, expectations, and objectives. Analysts allowed users to respond in their own words. Analysts did not limit answers to a narrow range.
	In-D15	Exchange Documents	Analysts exchanged the documents with the clients such as meeting agenda, requirement list, meeting minutes, and requirement review and requirement validation.
	In-D16	Future requirements	Analysts identified possible requirements for the future business processes.
	In-D17	Discussion of Client's objectives	Analysts identified the client's objectives.
	In-D18	Requirement reviews	Analysts checked the requirements document for inconsistencies and errors. Conflicts, contradictions, and errors in the requirements had been discovered by reviewers and formally recorded in the review report.
	In-D19	Discover user past experience	Analysts defined a user past experience that could be considered as a main issue of the effective interaction.
	In-D20	Facilitator	A facilitator helped the group work together productively to reach the best possible conclusions or decisions. The facilitator kept the discussion rolling, kept it on track, and summarized the discussed topics and issues.
	In-D21	Discuss meeting	Analysts discussed the main topic for the

		agenda	meeting.
	In-D22	Discuss progress since last meeting	Analysts discussed the progress achieved.

In-depth Description

ID	Paraphrased from interpretive data
In-D1	Build relation: One of the strategies in this stage was to build a relation with the client in order to make the client feel confident during discussions. This reduced tension and helped the client to easily express their thoughts.
In-D2	Explanation tool: Via the explanation tool strategy, analysts used different tools to help them explain the functionality of the software and its process. The tools used included a white board, blank paper, and diagrams.
In-D3	Body language: Body language is an important strategy used during the building of in-depth understanding. This suggested that analysts used body language to send messages to the clients. For example, gesture could be used during the explanation and presentation of the software. Body language also helped the analysts realize the level of understanding between themselves and the clients. For instance, the facial expression of the client could show if the client understood the need for a functionality of the software and if it was required.
In-D4	In-depth discussion: In-depth discussion helped the analysts get the details of the requirements and the client's business process. This in turn linked the gathered information with the related requirements. Collecting more details, like data about the requirements, about sources of the data, and about the users' involvement improved the potential product functionalities.
In-D5	Requirement validation: Analysts had written a requirement list to discuss with the users in order to validate users' needs. Analysts sometimes represented these requirements by prototyping screen. During the requirement validation meeting analysts and users discussed the requirement list and the possible modifications.
In-D6	Understanding database structure: Analysts tried to understand the client's data process within the requirements and the possible field in the database that covered this data process. This was one of the most requested functionality report mechanisms, requested by clients so clients figured out that data related to each single report that the analysts tried to understand.
In-D7	Identifying possible improvements: Through identifying possible improvements with the client, the analysts got a chance to add, improve, and modify the final potential product functionalities in order to fit with the client's expectation.
In-D8	Discussion of client's issues: By discussing the client's issues, the analysts gained a better understanding of the client's business process. They could also identify the challenges and issues that the users faced, and made sure they plan a potential product that would meet requirements and include the functionalities that would help the users deal with such challenges.
In-D9	Acquire knowledge: Before any discussion with the client the analysts went through the process of acquiring knowledge about the client's business process, objectives, and requirements. This was achieved by gathering general information about the business, gathering knowledge of the client's business area, and reading about specific business related processes on the company's website, for example.
In-D10	Clarifying information: The strategy of Clarifying information helped the analysts make the gathered and studied information clear by communicating with the client, asking questions related to their business process and software requirements.
In-D11	Define users' roles and Key Users: By defining the key users and the users' roles, analysts understood each user's involvement in the software and their level of interaction with the software. This helped the analysts to understand what each user required from the software, and helped assist when planning the restriction of some functions for particular users.
In-D12	Explain business process: Through communication with the client, the analysts clarified the business process of the organization. The clients and users explained the business process

	and activities of the organization in detail, which helped the analysts, discover ways to optimize such processes via the software.
In-D13	Get confirmation of process: After gathering the information and obtaining knowledge about the field and the business process of the organization, analysts used the confirmation of process strategy in order to confirm the information collected prior to their communications and discussions with the client. In this way they identified the correct requirements for the potential product.
In-D14	Open questions: Analysts used open questions in order to encourage the users to voluntarily take part in communication acts and discussions. In most cases the analysts tried to avoid closed questions that would lead to yes/no answers. The drawback of this approach is that it might bring unrelated issues and irrelevant information into the discussion. This, however, helped analysts build a friendly relation with the client, which in turn boosted client's confidence when it came to discussing related issues and helping in the development of a potential product. Open questions like "Could you tell me more about X?" allowed the clients to respond in their own words. This avoided limiting answers to the narrow range of choices that would result from asking closed questions.
In-D15	Exchange documents: During the exchange documents strategy, The documents included previous meeting minutes, lists of activities and topics for discussion. This documentation helped both analysts and clients to keep track of the software development process, and also provided boundaries for group discussions or meetings. In turn, this helped to direct focus toward those specific areas of information that analysts required clarification of.
In-D16	Future requirements: By applying the future requirements strategy, the analysts laid the path of future work with the existing client, but meanwhile put aside the further possible development of the software.
In-D17	Discussion of client's objectives: The discussion of client's objectives strategy helped identify the client's objectives and their expectations of the requested product.
In-D18	Requirement reviews: Both analysts and users manually checked the requirements documentation for any inconsistencies and errors. Any conflicts, contradictions, or blunders appearing in the requirements were pointed out by the reviewers and formally recorded in the review report. After the identified issues were recorded, it was up to the users, the system procurer, or the system developer to negotiate a solution to solve the problem areas.
In-D19	Discover users' past experience: Analysts discovered the users' experience with the current software (if they have any such experience) and the business process related to it. This helped with identifying the primary user who could be the main informant regarding the requirements for the product to be developed. However, the users' ability to create the requirements for the new software or to express their thoughts on the requirements could negatively impact on the requirement determination process.
In-D20	Facilitator: One of the members from the analysts' team acted in the role of a facilitator in order to help the group work together productively to reach the best possible conclusions or decisions. The facilitator handled the practicalities and logistics of the group's meetings, including setting the date and time of the meetings, reserving meeting space, etc. The facilitator's role was to keep the discussion rolling and to keep it on track, summarizing the discussed topics and issues. He/she, however, stayed impartial and avoided putting ideas or opinions forward themselves, or encouraging the group towards a specific conclusion.
In-D21	Discuss meeting agenda: The analysts discussed the topics presented in the meetings.
In-D22	Discuss progress since last meeting: The analysts presented and discussed the progress achieved from the previous meetings.

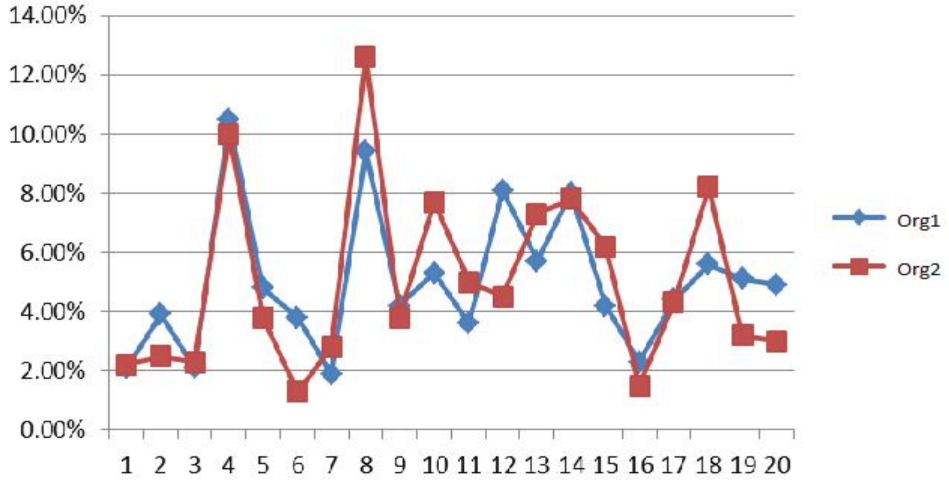
The table below shows percentage of inducted codes from organizations, during implementation, the inducted codes from Org1 ranged from 1.9% for identifying possible improvements to 10.5% for in-depth discussion. The inducted codes of building a relation and using body language were the second least, with a percentile inducted of 2.1% each. Future requirements did not have much higher inducted, with the result of 2.3%. The defining user roles and key users inducted code was at a level of 3.6%, while the understand database structure and explanation tools inducted codes had a very similar result of 3.8% and

3.9% respectively. The acquire knowledge and exchange documents strategies were inducted at 4.2% each. The discussion of client's objective (4.4%), requirement validation (4.8%), and facilitator (4.9%) all appeared within the fourth percentile rank. The inducted codes of discover user past experience (5.1%), clarify information (5.3%), requirement reviews (5.6%), and getting confirmation of process (5.7%) inducted from Org1. The open questions strategy and explain business process inducted code scored similarly, with 8% and 8.1% respectively. The second most commonly inducted code by Org1 was discussion of client's issues (9.4%).

During Implementation inducted codes

In-depth understanding	No.	ID	Org1	ID	Org2
	1	In-D7	1.90%	In-D6	1.30%
	2	In-D3	2.10%	In-D16	1.50%
	3	In-D1	2.10%	In-D1	2.20%
	4	In-D16	2.30%	In-D3	2.30%
	5	In-D11	3.60%	In-D2	2.50%
	6	In-D6	3.80%	In-D7	2.80%
	7	In-D2	3.90%	In-D20	3.00%
	8	In-D9	4.20%	In-D19	3.20%
	9	In-D15	4.20%	In-D9	3.80%
	10	In-D17	4.40%	In-D5	3.80%
	11	In-D5	4.80%	In-D17	4.30%
	12	In-D20	4.90%	In-D12	4.50%
	13	In-D19	5.10%	In-D11	5.00%
	14	In-D21	5.20%	In-D15	6.20%
	15	In-D10	5.30%	In-D21	6.30%
	16	In-D22	5.50%	In-D22	6.40%
	17	In-D18	5.60%	In-D13	7.30%
	18	In-D13	5.70%	In-D10	7.70%
	19	In-D14	8.00%	In-D14	7.80%
	20	In-D12	8.10%	In-D18	8.20%
	21	In-D8	9.40%	In-D4	10.00%
	22	In-D4	10.50%	In-D8	12.60%

The range of inducted coded for Org2 was wider than for Org1 and spread from 1.3% for the understand database structure to 12.6% for the discussion of client's issues. The second most inducted code from Org2 during the in-depth understanding phase was in-depth discussion (10%), followed by requirement reviews (8.2%). The open questions strategy and clarify information gained similar results, with 7.8% and 7.7%. The get confirmation of process inducted code was used 7.3% by Org2, while the exchange documents inducted code appeared in the next percentile down – 6.2%. The define users roles and key users and the explain business process had a difference of only .5% in their representation, i.e. define users roles, Key users at exactly 5%, while the explain business process was 4.5% of the time. Meanwhile, discussion of client's objectives was 4.3%, and both requirement validation and acquire knowledge inducted codes were used 3.8%. There was a difference of .2% between discover user past experience (3.2%) and the facilitator inducted code (3%), and then another 2% difference again to identify possible improvements (2.8%). After this, there were further small gaps between these occurrences and the representation of the explanation tools inducted code (2.5%), and the use of body language (2.3%). The build a relation was 2.2%, while the future requirements strategy came second to last at 1.5%.



As supported by the results from the table, build a relation, body language, and in-depth discussion were inducted at approximately the same level by both Org1 and Org2. Org2' least commonly inducted code was understand database structure; however, Org1 had a higher inducted codes than did Org2 for all three of the following codes: explanation of tools, requirement validation, and understand database structure. Org2 inducted identify possible improvements slightly more than did Org1, and discussion of client's issues reached its peak for Org2, leaving the second highest percentile inducted for the same code in Org1. While acquire knowledge and open questions codes were inducted at about the same level by both companies, Org2 inducted clarify information, define users roles, Key users, and get confirmation of process more than did Org1, which inducted explain business process strategy more often than Org2. While Org2 inducted exchange documents codes more often than Org1 did, future requirements code was less commonly inducted. Both companies applied the discussion of client's objectives code at nearly the same level. While Org2 inducted requirement reviews at a significantly higher level than did Org1, both the discover user past experience and facilitator codes were inducted more by Org1 than by Org2.

During Implementation inducted codes

In-depth understanding	ID	codes	Paraphrased from interpretive data
	In-D23	Meeting arrangement	Arranging the meetings Confirming the meetings
	In-D24	Validation and clarification questions	Validating and confirming the information previous gained
	In-D25	Document exchange	Exchanging the documents
	In-D26	Confirmation	Meeting confirmation Progress confirmation
	In-D27	Official document exchange	Presenting the official documents for signature
	In-D28	Closed questions	Yes/no questions
	In-D29	Building relation	Building relationship with the client

During Implementation inducted codes

ID	Paraphrased from interpretive data
In-D23	The analysts emailed the clients with information related to meeting arrangement. This included confirming the suitable data, time, and place for the meeting.
In-D24	After the analysts gained some information about the clients business process, they

	sometimes nevertheless had unclear information regarding the clients' business, for example, information relating to the organisation structure, the number of employees across different departments, the numbers of users for the developed software, the restricted access necessary for some of the users, and so on. This information was clarified by simple questions asked in an Email.
In-D25	The analysts provided the client with information related to the meeting agenda, and meeting minutes.
In-D26	After arranging demonstration presentation meetings with the client, the analysts confirmed the suitability of the meeting with the client' schedule, reminding them of the meeting's date, time, and place. The Emails confirmed further stages of the software development, in order to help the client to be able to follow the process. The Emails confirmed information related to the software development progress in order to fit in with the clients' deadlines.
In-D27	The analysts sent the product offer and product order for the clients to study and sign if agreed with. The initial requirements were also sent to the clients to study and sign if approved of.
In-D28	The analysts emailed closed yes/no questions to the clients in order to gain more information related to the software development.
In-D29	Emails sent by the analysts contained icons to represent smiles (e.g. ☺) which brought the Emails to a friendly level, helping to build better relations with the client.

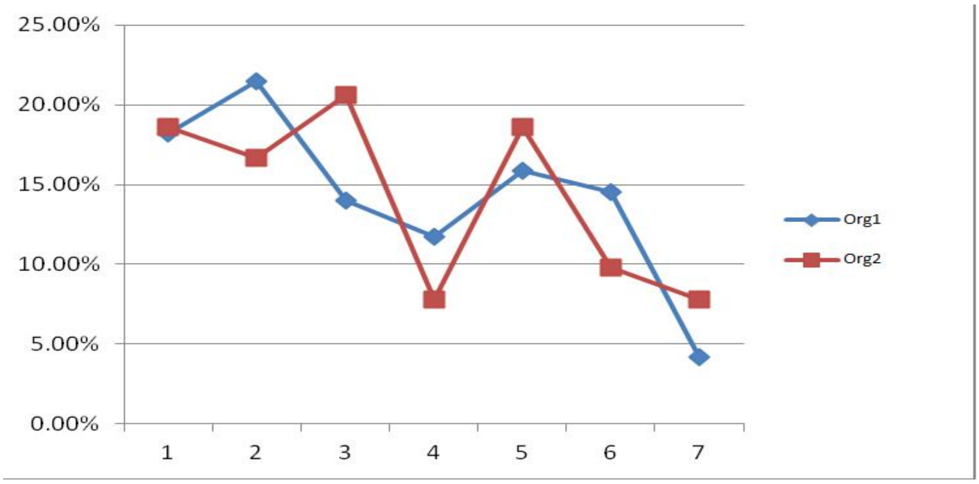
During Implementation inducted codes

In-depth understanding	ID	Org1	ID	Org2
	In-D29	4.20%	In-D26	7.80%
	In-D26	11.70%	In-D29	7.80%
	In-D25	14.00%	In-D28	9.80%
	In-D28	14.50%	In-D24	16.70%
	In-D27	15.90%	In-D23	18.60%
	In-D23	18.20%	In-D27	18.60%
	In-D24	21.50%	In-D25	20.60%

According to the results from the table above, during implementation, Organisation 1 inducted validation and clarification questions 21.5% of the time. Organisation 1's second most inducted code was meeting arrangement (18.2%), followed by official document exchange (15.9%). With Organisation 1, closed questions and document exchange codes had a gap of .5% between each other: closed questions code was inducted 14.5% and document exchange was inducted 14%. The second to last code was confirmation (11.7%). There was then a large percentile gap to the least inducted code– building relation, the inducted for which was only 4.2%.

The inducted of the same codes in Organisation 2 ranged from 20.6% for document exchange to 7.8% for both confirmation and building relation codes. Meeting arrangement and official document exchange were inducted 18.6% each, followed by validation and clarification questions (16.7%). The second-least used code was closed questions (9.8%).

As the figure below represents, both companies inducted meeting arrangement at about the same level. Clearly Organisation 2 paid relatedly inducted less such codes as validation and clarification questions, confirmation, and closed questions. On the other hand, such codes as document exchange, official document exchange, and building relation were inducted at a higher level by Organisation 2 than by Organisation 1.



It is a long process. Listen, first, we have our marketing team who call a lot of companies to find out if they are interested in any type of software that we provide….. We keep updating our database about potential clients. We got this information from newspapers, searches on the internet, different ministry websites such as ‘education’, ‘manufactory’ and so on… The important thing for us is to find out if they are looking for a software product to use. This of course depends on their kind of business… so we try to advertise our software product to them… most of the time they ask to send information about our software product so we tell the sales team to visit them…. Before such a visit happens we try to get as much information as possible about the potential clients, such as their business type. Formal Interview (07/01/2012)

Client Name: A
Address: AAA
Demonstration Request: Yes
Time and Date:
Request From: G
Client importance: high
Place:
Client Description: Client A. is a primary and high school that includes 60 employees and 3000 students. A. has about 21 users of the packaged software “School Management System (SMS)”. Client A. has 5 departments that use SMS: Students’ information, School clinic, School Library, School enrolment, and Financial department.
Client A. used a SMS at their school. However, Client A. is interested in replacing their SMS. The reason for this is that the current SMS does not support the central database. In other words, the five departments are not connected with each other through the SMS so they normally send their transactions relating to student profiles manually via paper documentation.
[Demonstration Request Form Information]
The customer analysis form, client’s company structure information and software demonstration request information help us to decide on software demonstration mechanisms... you know, it helps us decide what different issues impact on our choice and how the client’s type impacts on our choice… for example: the number of users within a company lets us decide if we are going to demonstrate a VB.NET solution that is more expensive and serves a greater number of users, or a VB6 Solution that is less expensive and can serve only a limited number of users…sometimes clients have issues outside our work domain, so we just cancel the software demonstration… informal Interview (08/01/2012)

During Implementation inducted codes

Pre-implementation	ID	codes	Paraphrased from interpretive data
	I-D11	looking for potential clients	marketing and sales looking for potential clients on a daily basis
	I-D12	record status of potential	marketing and sales keep on following up

		clients	with potential clients
	I-D13	Record potential clients information	Marketing created an internal database resource that provides information about clients
	I-D14	Arrange a meeting for sales team	Marketing team usually begin the interaction with potential clients through phone calls and emails in order to arrange a meeting between them and the software sales team.
	I-D15	Receive requests for software demonstrations	Analysts received a software demonstration request via their sales and marketing teams.
	I-D16	Uses technical language	Sales and marketing teams provide a technical issues description about potential clients' organization
	I-D17	response to a software demonstration	Analysts asked the clients about specific features needed in the software Analysts discussed possibilities of improving, adding, or modifying some functions for the client's satisfaction.
	I-D18	Discussion of client's issues	Analysts identified client's issues within their business process Analysts tackled the issues by changing the software.
	I-D19	supports the client's business	Analysts explain the software in term of client business in order to convince the client to buy the software
	I-D20	present the key functions	To show the software benefits to the client's business by stress what is important for them.
	I-D21	presenting a possible solution	Analysts try to present a possible solution to the clients by use the demonstration request
	I-D22	initially perceived	Analysts demonstrate some features, the client may decide on some new requirements, thinking those requirements very attractive when they did not initially consider them to be requirements. Analysts will perceive even these new requirements as part of the initial requirements.
	I-D23	real case scenario	To simulate and cover various aspects of a real situation within the client's work environment
	I-D24	time limit	Constraints on the time allowed for software demonstrations made analysts concentrate on very specific client issues during demonstrations.

Appendix C: Define the events and situation of PSIRE

The first one and half months, during which I was trying to understand what process analysts apply, helped me build a trust relation with the participants. I have socially interacted with participants during coffee breaks or lunch time, later I have played several soccer games with them and attended two wedding parties of the participants.

In undertaking field observation, I have started to observe the events that happened in both organizations in terms of PSI and I have recorded these events by taking notes as following:

> Site: org1
> Data collector: Issam
> Date: 8-2-2012
> Start: 1 PM.
> When I arrived, a group of people came to the company and they have been asked nicely to have a seat in the meeting room. After that one of team leaders who was an analyst went inside and they stayed there for 2 hours. This kind of thing had been happenning many times and made me interested in terms of if it is related to requirement determination.
> Site: org2
> Data collector: Issam
> Date: 7-2-2012
> Start: 9 AM.
> When I was at my desk, I went first to the analysts' project room where several analysts were sitting. I have asked about one of them and the answer was that he was outside because he had a demonstration presentation. This kind of thing had been happenning many times and made me interested in terms of if it is related to requirement determination.

At this point, I have started to investigate about the demonstration presentation meetings and if it is related to requirement determination. I have discussed this with the participants.

> I: Hey mate, How is going?
> Participant: I am good and you?
> I: I am good too. Listen, I would like to ask you about those people who are coming here for demonstration presentation in the meeting room.
> Participant: There are the clients
> I: Really? That's impressive. How do you bring them here?
> Participant: It is a long process. Listen, first, we have the marking team who is calling a lot of companies to find out if they are interested in any type of software that we provide.

Consequently, I have found that marking team always begin the interaction with potential clients through phone calls and emails in order to arrange a meeting for the product sales team. The discussion of demonstration presentation has been continuing when the meeting is appointed.

> I: Woow that is cool, and what if they are interested?
> Participant: Then sales team visit them and try to get more information about what they are interested in and they invite the client to attend a demonstration presentation.

The sales team then met with the client in order to explain more details of the software products that their company could provide. At these meetings, sales team collected initial information about the potential client's company structure. After this, the sales team arranged a demonstration presentation about the products; this demonstration was carried out by the analysts. The sales team then completed meeting minutes and a customer analysis form that were used by analysts to prepare for their demonstration of potential products. The discussion of demonstration presentation situation has been continuing.

> I: Who is doing the demonstration presentation?
> Participant: The analysts team leaders.

Analysts then carried out the demonstration presentation about the potential products. At this pointed, the demonstration presentation situation was an interesting event for me since it had been applied many times by both organizations and analysts were responsible to carry it out.

In order to understanding the purpose behind the demonstration presentation, I have requested to attend some sessions of demonstration presentation to get more information. These sessions had been noted and tape recorded (wherever possible). I have started observing the events occurring during demonstration presentation situation. In addition, I had asked analysts' team leaders for interviews since they were responsible to carry out the demonstration presentation.

I have asked myself about what happened after demonstration presentations? To answer that question, I had asked more questions and took more notes to figure that out. I found out that analysts created a report about the client's initial needs. This report was later presented to the analysts' head management in order to price the product. The report showed initial needs of the client, the analysts' expectations of the development time-frame involved, and comments relating to possible ways to improve the potential products in order to fit with the client's initial needs. It was clear that the process structure had been applied by both originations analysts team.

> They are never convinced about the product until they see what product functions are and the outcome, such as report and invoice - Participant - Formal interview
> You know what? Everything is about business. I pay and you have to convince me about your product so we make a live demo to users to show how good we are - Participant - Informal Interview
> At the demonstration presentation, we collect initial requests from the client and then we write a report to the manager in order to price the product - Participant- Informal Interview
> Demonstration presentation helps us know the clients' initial needs - Participant - Informal Interview
> We have to know what client issues are to provide a good solution because sometimes client doesn't know what the right solution is so we make a demo to know that first - Participant - Informal Interview

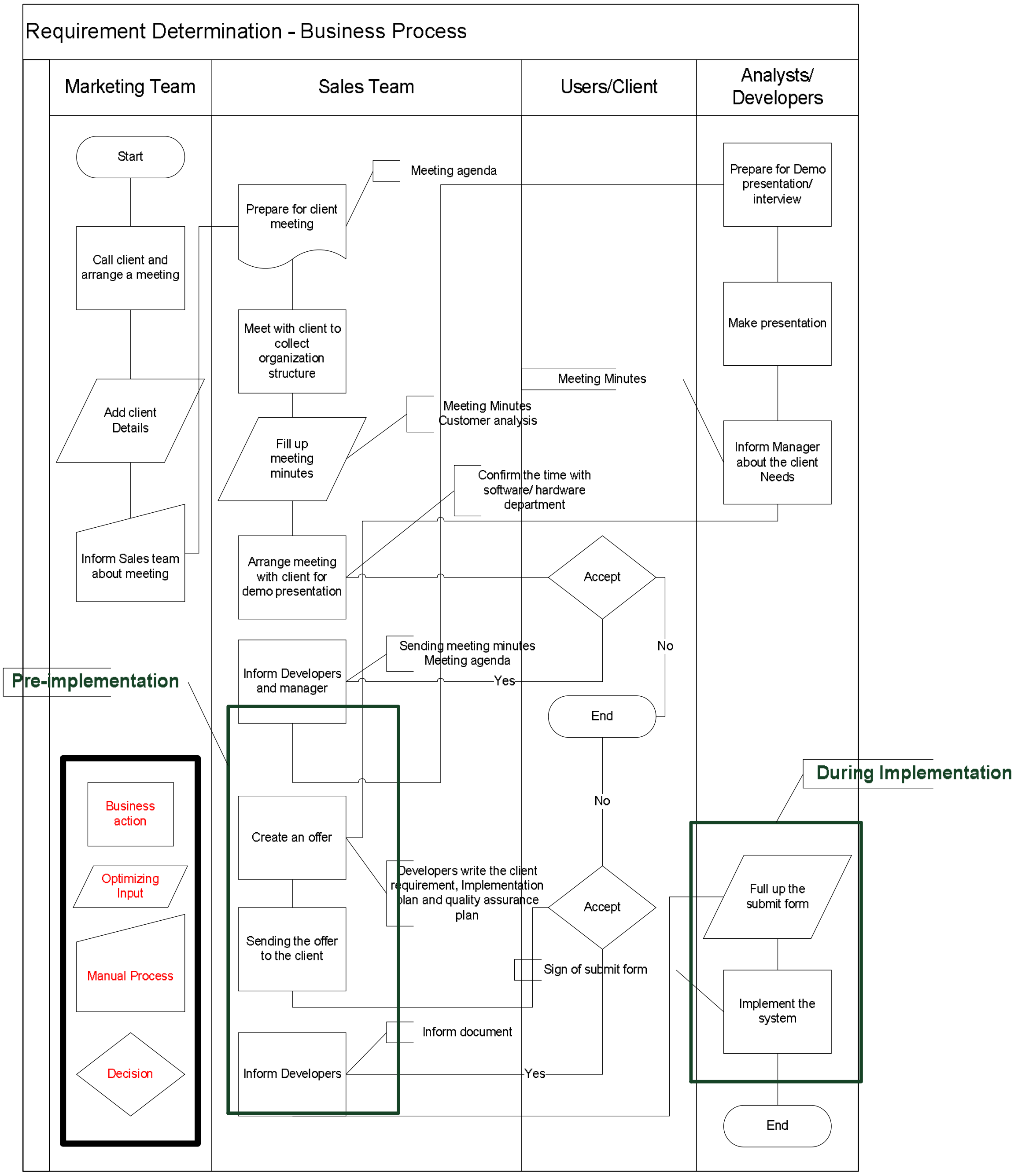
In case the client accepts a product offer that has been provided by the head management of the company, the in-depth gathering of information about the client's needs follows.

> Site: org1
> Data collector: Issam
> Date: 12-2-2012
> Start: 9 AM.
> When I was at my desk, analysts were preparing to go and meet with the client. The meeting was held at the client venue. I have found that there was a schedule for meetings that analysts went to.

This was an interesting that analysts were meeting with the client to collect more information about his/her needs.

> Observation note: "Analysts team prepare for meeting with users to collect more requirement description"
> "After product offer is accepted by the client, we use the information in initial report to start collect more detail about client needs" USER2- Formal Interview (23/03/2012)
> "At demonstration presentation we got high level of client's needs but still we have to get more detail about these initial needs" USER1- Formal Interview (23/03/2012)
> "Well, we know what client's issues are from the demo meeting so we use these issues list to get more explanation from client. Anyway, it is all about solving client's issues and this needs more discussion and understanding to his business process" USER5- Formal Interview (23/03/2012)

I have drawn the organization business process flow chart in order to identify when requirement determination occurs. I have met each team involved in this process. The figure below shows the organization business process from the participants' perspective. I have highlighted when analysts and users were involved and requirement determination occurred.


Requirement Determination - Business Process
Marketing Team
Sales Team
Users/Client
Analysts/ Developers
Start
Call client and arrange a meeting
Add client Details
Inform Sales team about meeting
Prepare for client meeting
Meeting agenda
Meet with client to collect organization structure
Fill up meeting minutes
Meeting Minutes Customer analysis
Arrange meeting with client for demo presentation
Confirm the time with software/ hardware department
Inform Developers and manager
Sending meeting minutes Meeting agenda
Create an offer
Sending the offer to the client
Developers write the client requirement, Implementation plan and quality assurance plan
Inform Developers
Inform document
Meeting Minutes
Accept
No
Yes
End
No
Accept
Sign of submit form
Yes
Prepare for Demo presentation/ interview
Make presentation
Inform Manager about the client Needs
Full up the submit form
Implement the system
End
Pre-implementation
During Implementation
Business action
Optimizing Input
Manual Process
Decision

List of analysts' Practices in context of PSI

Situation (where)	**Process (when)**	**Practice (How)**
Pre-Implementation	Demonstration presentation	Explaining product functionality Initial explanation Uses the sales team report live scenario Link the product functions to business process Discuss the business process
	Create Offer	Software Offer Elements
	Search for Potential Clients	Through phone calls and emails in order to arrange a meeting for the product sales team Explain more details of the software products that their company could provide
	Software Demonstration Request form	Prepare for their demonstration of potential products
	Response to Software Demonstration Request	
	Discussion of Packaged Software Offer with Client	
Situation (where)	**Process (when)**	**Practice (How)**
During-Implementation	Installation	Technical needs assessments Installing the copy of the software
	Software Demonstration, identify mismatch	Respond to the discovery of a mismatch Explaining product functionality Identify the mismatch between software functions and users' needs

Appendix D: Day by day analysis

Feb 1, 2012 - Organization 1- Pre – Implementation/ Software demonstration			
Events	Pre – Implementation/ Software demonstration	**Actors**	User2
Diary Observation (translation, notes)	• Creating a live scenario for the software demonstration • The client had some issues with their existing software, as noted in the sales team report, such as their inability to follow the order, missing orders, and difficulties with the inventory of items. • Analysts explained what the software could do in order to solve the client's issues. • Analysts linked the product functions to business process by explaining some of the business process cases to show how product functions cover the client's business.		
Meeting Notes	**Pre - Implementation**	**Organization 1**	**Inventory and purchases**
	• Analyst Clarify client issues at the start of the meeting. • Analyst uses the sales team report to ask about the client issues. • Analyst explains that the software will help to solve his issues. • Analyst presents the software main parts. • Analyst start by the setting functions for the software to be running. • Analyst goes through a software functions • Analyst explains the software functions such as add new item, items level of categorization. • Analyst discusses the inventory business process • Analysts uses a real case to explain the software functions such as add item and make item order through the software.		
\Descriptive analysis	I attended a software demonstration meeting during which the analysts aimed to show their product to potential clients and to convince them that the software product they had to offer would satisfy the client's needs. "We represent our software to a client by developing a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment. It really helps us to explain our software functions and connect these functions to a real case [team leader]". The clients run a goods inventory and they currently have some software that they use in the company. However, their current software is not adequate for their needs and it currently has some issues. This causes problems like the staff of the inventory having problems following an order, missing orders, and difficulties relating to the inventory of items. All of these issues had been made known to the analysts through the sales team report. The analysts went into the meeting with the purpose of telling the clients about their own inventory software which they think will solve the client's issues. The analysts did a few different things during this meeting. At the start of the meeting, first they clarified the client's issues. They kept referring back to the sales team report in order to ask about the client's issues. They then began explaining what their software could do to solve those issues faced by the client. They gave an initial description of their software product's functionality. It seemed that they did this in order to give the clients some kind of vision about the product's functions, and in order to increase the client's participation in the discussion. But the main thing that the analysts did to try to convince the client that their product was the right answer for them was to incorporate a live scenario to use during the software presentation. So they didn't just describe the product and its functions but actually demonstrated it live for the client, giving it a kind of test case and showing that it could do the work the client needed. This meant the analysts actually went into the client's inventory and began processing items orders using the software they had made. Actually, this live scenario seemed to have two different purposes: one was to try to convince the client to buy the product, but the other was also to use the live scenario to help analysts collect more information about client needs. For example, while using the software in a live scenario, they could show the clients the software while it was actually running, but they could also gain a better immediate understanding of the problems the client was having – and then become better informed about how to identify possible solutions. The analysts first set up the software functions that needed to be running, then described the various software functions, and then demonstrated the software functions as they worked. This included showing the client features such as 'add new item' and showing them levels of categorization available. They then used a real case scenario of trying to place a items order so they could show how the software would let the client 'add item' and make an		

	item order. For example, they showed the client the interface for making items orders, and they showed the client what would happen. The client actually wanted to have a choice where particular items orders were sent to and therefore needed to have reports in the system. During the live scenario the analysts showed that the software had this capability. The whole time that the analysts demonstrated the software they kept on linking the product's functions to the client's business process by connecting some of the client's concerns about business process (i.e. things that the client said had gone wrong before and various things that they wanted the software to do) to what the software could do. They also discussed the client's inventory business process with the client. The end result was that they managed to show the client that the product functions could cover the client's business. There were people present on the analysts' team and. The people sent by the client were the manager of the inventory. During the whole time, the client seemed pretty happy or impressed by the software and at the end of the demonstration the analysts suggested that they thought the client would probably buy it. The tone of the meeting was very business-like and the analysts made sure to be really clear and efficient.			
Immediate interpretations of events	**Situation (where)**	**Process (when)**	**Practice (How)**	**Causal Explanations (why)**
	Pre-Implementation	Demonstration presentation	Explaining product functionality Initial explanation	• To give an initial explanation about the product. • To help clients create vision about the product functions, to increase clients' involvement and participation in discussion • To present the functionality of the software.
			Uses the sales team report	• To Clarify client issues • Explained what the software could do in order to solve the client's issues.
			live scenario	• To help enhance understanding of the problem, and simultaneously identify possible solutions. • to convince them that the software product they had to offer would satisfy the client's needs • to develop a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment
			Link the product functions to business process	• To present the functionality of the software and identify its need according to the business process.
			Discuss the business process	• To understand clients' needs and their business process

Feb 5, 2012 - Organization 1 - Pre – Implementation/ Software demonstration				
Events	Pre – Implementation/ Software demonstration		**Actors**	User4
Meeting Notes	**Pre- Software demonstration**	**Organization 1**	**Accounting Software**	
	• Analyst welcomed the clients and introduced himself. • Analyst presented the product name that would be discussed at that session. • Analyst started with technical elements of the product, such as database, number of users, the product platform etc. • Analyst explained the functions that product supported.			
Descriptive analysis	I attended another software demonstration meeting during which an analyst showed their software product to potential clients and tried to convince them that the product they had to offer would satisfy the client's needs. The software being presented this time was a kind of accounting software. As the analyst presented the product, he tried to provide an initial explanation about the product and to explain the product's functionality. The way that the analyst preceded was to first welcome the clients, then to introduce himself. He then mentioned the name of the product that was about to be discussed. Then he started explaining the product's functionality and explaining the functions that the product supported. He started with explaining the technical aspects of the product, such as the database or what it can do to support/ manage databases, the number or users it could support, and the product platform. As in the previous demonstration presentation, the point of the presentation was to try to give the client some vision of how the software product functioned and how it could work for them. The analyst also tried to encourage the client's participation in the discussion. Through asking questions and open discussion about accounting business process and functions for example, he asked the client questions if the product's functions matched what they wanted/ ask if they wanted more functions; ask about the client's business, What kinds of screens or interface did they show them, and the analyst refer back to the sales team report. The client's business is about second hand car parts. The client's issue was in tracking cheques payment and how could the software manage the payment through the cheques. Client asked if the software provides a reports about due cheques, if the software supports cheques return. I noticed that during the meeting, the atmosphere was pretty friendly/ relaxed. The client did not ask a lot of questions but seemed happy to sit quietly and let the analyst show them the product. The client seemed to be quite happy with what the product did and it appeared likely that they would purchase the product and the client seemed to express some doubts about the product After the demonstration, the analyst told me that he thought the demonstration had gone well.			
Immediate interpretations of events	**Situation (where)**	**Process (when)**	**Practice (How)**	**Causal Explanations (why)**
	Pre-Implementation	Demonstration presentation	Explaining product functionality Explaining the functions that the product supported Presented the product	• To give an initial explanation about the product. • To convince them that the product they had to offer would satisfy the client's needs • To explain the product's functionality • To try to give the client some vision of how the software product functioned and how it could work for them.
			Explaining the technical aspects	• to give view about software technical supportsd such as the database or what it can do to support/ manage databases, the number or users it could support, and the product

				platform
			Asking questions and open discussion about accounting business process	• To encourage the client's participation in the discussion
			Usage of sales team report	• To Clarify client issues • Explained what the software could do in order to solve the client's issues.

Feb 8, 2012 - Organization 2 - Pre – Implementation/ Software demonstration			
Events	Pre – Implementation/ Software demonstration	**Actors**	User5
Diary Observation (translation, notes)	• Trying to understand the client's issues and challenges helped the analysts discover the correct expectations for the software. • Evaluation of users' needs; in which consists estimate cost and time of customization effort for users' needs. • The analysts were of the view that the reason for discovering client's expectations is to understand the product scope. The reason is that to figure out client's expectations at the beginning helps the analysts understand what the client really wants, which, in its turn, helps them discuss an issue or a request with the client and then do the work. • Discovering client's expectations helped them define the project scope through discovering client's problems, needs, and deliverables. Define the project's scope is one of the key factors in requirement determination. • The analysts set up the priorities to the defined scope of the product, make decisions on what has to be complete and what should be left behind. • Some questions were related to the product functionalities that were mentioned in the presentation but in more depth mentioning some problems that the users seemed to have at the moment.		
Meeting Notes	**Pre - Software demonstration**	**Organization 2/ Client site**	**HR**
	• Analysts' discussion of the client's issues, • Identify the challenges and issues that the client was facing at that moment with their business or software which was in use at that point of time. • Analysts tried to assure that the potential product met the requirements for the client's business by discussing the client's issues • Identifying the client's requirements. • Analysts have explained employees bonuses mechanize that software provides and support but it was not suitable for users' bonuses mechanize process. • The users' bonuses mechanize was not much with the software function, • The provision of uniforms was not much • Output format issues		
Descriptive analysis	I attended a software demonstration that was held at the client's site. The discussion during this software demonstration was quite complex because the client had a lot of requirements and asked for changes to be made to the software that Organization 2 was offering. The analysts had to explain that their software already delivered some of those requirements without any changes needing to be made. The software that was demonstrated was a form of HR software and needed to deal with various payments and reimbursements made to employees of the client. Before we went to the software demonstration, the analysts told me that the purpose of their software demonstration was to clarify the expectations that the client had already mentioned they wanted from the software and better discover what the client really wanted. Therefore the meeting and demonstration would be used to understand the project scope. The analysts wanted to know this early on as it would help them to know what the client really wants. They thought that clarifying the whole project scope first would help them discuss specific issues with the client and then be more certain when doing the work. They felt that defining the project's scope would depend on their finding out about the client's problems, needs, and deliverables. They thought that defining the project scope was one of the key factors, and explained that once expectations were confirmed or clarified they could better discuss such requests with the client. They could also discuss any other issues relating to the software product and then discuss how the software company would go about doing any further work on the product. During the demonstration the analysts did a number of different things. They began by discussing the issues that the client was already hav ing in relation to the HR software that they were currently using. Problems were arising in relation to that software. The analysts asked various questions to make sure that they properly understood those issues and that they understood what challenges might arise and they tried to create the right software for the client. By having such a discussion they were able to discover the right requirements of		

	the software and could also have a more solid base for assuring the client that their software would be able to deliver what the client's business needed. The client has a number of serious problems with their current HR software and idiosyncratic needs of software due to the some of the specific practices within their business. Because if these specific needs, as the analysts demonstrated the product, during the demonstration the client actually asked for some further functions to be added. One issue that the client had was the need to have software set up in a specific way to deal with how their company's bonus mechanism worked. Although the analysts had already explained how the bonus mechanism provided by their software worked, it was found that it did not in fact match up with how bonus mechanisms worked within the client's company. Therefore, the client asked that this could be changed. There was also a mismatch when it came to the mechanism that related to the client company's provision of staff uniforms and reimbursement for the cost of uniforms that staff members had originally had to cover "80 JD is taken from the first month of an employees' salary; return the 80 JD after 3 months if the employee is still active or minimum working days is 70". The analysts explained that making the bonus mechanism work the way that the client preferred would require customization, which would involve cost and time. They told the client that they would get back to them with a report that would show the estimated cost and time or making such a change. The comments and questions that the client directed toward the analysts were sometimes related to the product functionalities that the analysts were presenting, but more often were related to the problems that the client was actually facing at the moment – and this led to quite a lot of in-depth discussion of those problems. So, as the meeting developed, the purpose of the demonstration became not only to demonstrate the product and try to sell it, but to start estimating the time that would be required to engage in such customizations as the client wanted, and to estimate the cost of such a customization effort. After the demonstration the analysts put together a report explaining what such extra costs would be. I could see that part of the job of the analysts during this meeting was to start helping the client deciding on priorities for the product, and that the analysts had to start making decisions about what customizations should be completed and what customization requests should be left behind. It seemed that these decisions were mostly controlled by the client's wants, but customizations would still only occur if the client was ready to agree to the time and costs that would be involved.			
Immediate interpretations of events	**Situation (where)**	**Process (when)**	**Practice (How)**	**Causal Explanations (why)**
	Pre-Implementation	Demonstration presentation	Discussing the client's issues	• To clarify the expectations that the client had already mentioned they wanted from the software and better discover what the client really wanted
			Clarifying the whole project scope	• To understand what the client really wants which helps them discuss an issue or request with the client and then do the work. • To understand the project scope • To define client's problems, needs, and deliverables.
			Ask various questions about client's business process	• To make sure that they properly understood those issues and that they understood what challenges might arise and they tried to create the right software for the client. • To discover the right requirements of the software and could also have a more solid base for assuring the client that their software would be able to deliver what the client's business needed

			Actual misalignment	• Making the bonus mechanism work the way that the client preferred would require customization, which would involve cost and time.
			Evaluation of users' needs	• To consists estimate cost and time of customization effort for users' needs. • provide a report that explain the extra cost of these users' needs

Feb 9, 2012/ Organization 2 - Software Offer				
Events	Software offer		**Actors**	User5 and User6
Diary Observation (translation, notes)				
Meeting Notes				
Descriptive analysis	Today I learned about how the software analysts actually go about deciding how to create a software offer. I was present at the meeting at which they discussed creating an offer, and they explained to me what some of the most important factors to consider were. The reasons for meeting to discuss what elements should go into the software offer were that the analysts needed to define the software scope offer; the analysts needed to create and provide to the client clear guidelines regarding which pre-conditions had to be met for a packaged software implementation to go ahead; and the analysts needed to be able to work towards accurately estimating the time and cost required for the software implementation. “It was a big issue, and it had required a lot of effort and time to work with that undefined scope to understand business practices and business requirements. Therefore, our strategy now is to define the scope of the software early on so that we will be prepared for the next step in the case that clients accept the software offer” [User5]. “If the client already has software and he /she needs to replace it, there will definitely be some functions that our current software does not provide, so we need to know what these functions are and then assess whether we can provide them, otherwise the client will not buy our software since we could not provide solutions for his/her issues” [User6] I discovered that the different elements that went into the software offer included the following: • Identifying the initial requirements for the software. These could also be considered as ‘core requirements’. In practice, these requirements include high level modification requirements and the addition of new features. In this particular case, the new features to be added related to transaction functions in the packaged software. The HR software being created in this instance needed to feature transaction functions that could deal with an employees’ bonus mechanism and with reimbursement for uniform costs. • Software output forms which allow the client to make a number of modifications to packaged software, such as changing a report format output. • Technical dimensions: these technical dimensions could relate to features of the client’s infrastructure, such as their server and their software program.				
Immediate interpretations of events	**Situation (where)**	**Process (when)**	**Practice (How)**	**Causal Explanations (why)**
	Pre-Implementation	Create Offer	Usage of Software Offer Elements	• To define the software scope offer • To give clear guidelines regarding which pre-conditions have to be fulfilled for a packaged software implementation to go ahead. • To work towards its main goals of accurately estimating the cost and time required for packaged software implementation • To know what these functions are and then assess whether we can provide them

Feb 26, 2012 - Organization 2 – Software demonstration, identify users' needs, validation of users' needs			
Events	Identify clients issues with the software	**Actors**	User5
Diary Observation (translation, notes)	• Since the client had accepted the software offer, further in-depth understanding of the users' needs was required. • Analysts discussed such needs with users, and more description of users' needs was gathered • Using the printouts of the software function screens to add users' needs is a strategy that analysts utilise during the requirements validation practice for packaged software implementation. • Graphical representation of the user requirements, user environment, and organization department structure through diagrams showing the relations between departments helped to describe the user environment. • My observations of analysts showed me that identifying mismatches was an important process that was applied in most cases when analysts offered products to users. In this case I observed, analysts tried to get details of the users' business process and the requirements related to the software scope.		
Meeting Notes	**Du-Software demonstration**	**Organization 2/ Client site**	**HR**
	• There is no output and input change needs • Change the transaction formula relating to providing a uniform • Analysts discuss the client business process for providing a uniform • Analysts used a print screen of the software to validate the client needs • Sign of the change request regarding providing a uniform formula. • Analysts train users on other functions of HR • Analysts explain the report list provides by the software. • Graphical representation of the client's organization departments' structure and diagrams showing how they are supported by the software • Analysts sign the users' meeting summary. • Manager asked the analysts to add some attributes into employees' salary reports • Analysts explained to the accounting manager that the attributes requested already exist in the software		
Descriptive analysis	Today I accompanied the analysts on a trip to the client's company site. The analysts engaged in a software demonstration of the HR software they had developed for the client. The client had already accepted the software offer made by Organization 2, but now a more in-depth understanding of the users' needs was required. I observed that there seemed to be various reasons behind the meeting/demonstration: some were to do with the software itself, some were to do with the client's requirements, and some were to do with the client's company environment. As the analysts showed the client the software and discussed the client's needs from the software, it was apparent that the analysts were trying to validate the client's needs (i.e. confirm those needs/requirements they had already collected), to get a better understanding of the client's business process, and to identify any mismatches between the client's needs and the software. These actions helped the analysts know more about the client's requirements. While showing the client the software, the analysts also engaged in explaining different elements of the software. They explained its various functionalities and clarified various textual and graphical materials. The analysts tried to demonstrate the software in a way that showed how it could support the client's business. They also trained some users at the client's organization on using some of the software functions. The analysts showed the client printouts of various software functions, in order to gather further client needs or gain more detail about the client needs they already knew about. They were therefore using a copy of the packaged software in order to explain the software functionalities, and in order to carry out requirements validation. The analysts also tried to make graphical representations of the users' requirements and of the users' (client's) environment. Graphical representations (i.e. sheet of paper) were made of the relationships between the organization's different departments. It appeared that this was done in order to get an understanding of how the software would have to help these departments interact or how it would need to serve all the departments. It looked like the analysts were trying to get an understanding of and then properly describe the users' environment. By showing the client all of these software functions and drawing diagrams of the client's user environment the analysts were able to get more information about the necessary software scope, the client's business process, and any mismatches remaining between the software and the client's business practice.		

	The main actions taken by the analysts during the software demonstration/meeting included the following: in relation to customization requests, showing/confirming that there were no output and input change needs, discussing the client's business process in relation to how they provided staff uniforms and reimbursed staff for the uniforms' purchase or return, and a discussion of the possibility of changing the transaction formula related to their provision of uniforms, and using a printed screenshot of the software to validate the client's needs; explaining the report list provided by the software; training users on other functions of the HR software; and creating a graphical representation of the client's organization structure and relationships between their departments. The analysts ended up signing off on the 'change request' that the clients had made regarding the formula related to uniforms. To me, it looked like there were two main benefits to having such a 'change request' form and needing to have it signed: it would be useful to have an official documented request to refer back to later, for information or for confirmation that an agreement was reached, and this request form helped to manage the whole implementation process. The analysts did turn down one customization change request, though. The Accounting Manager of the client organization then asked the analysts to add some attributes to the employee' salary reports. The analysts answered this by saying that the attributes that were being requested already existed within the software. At the end of the meeting, the analysts signed off the users' meeting summary. had a list of topics to discuss during the meeting, or a list of needs they wanted met, and when the analysts signed off, it meant they were saying they'd spoken about all those topics or agreed with all those requests.			
Immediate interpretations of events	**Situation (where)**	**Process (when)**	**Practice (How)**	**Causal Explanations (why)**
	During-Implementation	Software demonstration, identify users' needs, validation of users' needs	In-depth understanding of the users' needs Engaged in explaining different elements of the software	• To validate the client's needs (i.e. confirm those needs/requirements they had already collected), • To get a better understanding of the client's business process • To identify any mismatches between the client's needs and the software • To demonstrate the software in a way that showed how it could support the client's business
			Identify clients issues with the software	• To validate the client needs • To understand the clients' business process • To identify the mismatch between client needs and software
			Sign off change request	• To manage the process • To have an official record of the change request
			Explanation tools	• To help the analysts explain the functionality of the software and its process. • To clarify the textual and graphical materials • To help enclose and describe the users' environment, which assists in clarifying and specifying requirements understanding.
			Demonstrate &	• To train users about the

			train users	software functions. • To demonstrate how the software supports the users' business

March 7, 2012 - Organization 2- Pre – Implementation/ Software demonstration			
Events	Pre – Implementation/ Software demonstration	**Actors**	User5, User6
Diary Observation (translation, notes)	• Preparing for the demonstration presentation by using the sales and marketing team's report about the client's organisation structure and the initial issues involved. • Creating a live scenario for the software demonstration • The team of analysts then sought to represent these functionalities by using a live scenario that allowed them to link software functionalities to the business case. • The team leader of the analysts explained to me that creating a live scenario of a business case through software demonstrations was usually capable of delivering the possible solution that the client needs. • The questions that analysts asked the client were related to linking the product functionalities with the client's specific business needs. • In the case that a function was not fully supporting the client's business, the analyst was making notes of it and agreeing to add the function or modify it at a later stage. • The analyst clarified and confirmed with the client his understanding of the example given by the client relating to which function/s needed modification.		
Meeting Notes	**Pre – Implementation**	**Organization 2**	**H2O**
	• Analyst clarifies client issues at the start of the meeting. • Analyst uses the sales team report to ask about the client issues. • Analyst explains that the software will help to solve those issues. • Client's issues are inability to follow the order, missing orders in the kitchen, and difficulties with the inventory of items. • Analyst presents the software main parts. • Analyst start by setting the functions for the software to run. • Software includes hardware functionalities, such as using a personal digital assistant (PDA) to take customer orders, and software functionalities. • Explained what the software could do in order to solve the client's issues. • Listed a set of functions that the software provided, such as making it easy to take an order, making sure the kitchen receives the order, helping the cashier to receive the payment required, and creating a record of their being paid • The client had two different types of menu: one for local customers and another for tourists • The client also had three types of service: takeaway, delivery, and internet service		
Descriptive analysis	Today I attended with the analysts software demonstration for H2O software "restaurant management system", as the analysts had decided to demonstrate their software to the client by using a live case scenario. This means that they were going to show the client the software running in real-time, and in the location in which it would be used. From what the analysts said, there were a few reasons behind choosing to do a live case scenario: it was a good way of presenting the software and being able to explain the product functionality; it was also a good way of clarifying what the client really needed and gaining more information that would help them deliver a suitable product, since they would see the client's work environment; since using a real case scenario simulates a real situation, it is effective in convincing a client that a product's functions are suitable – i.e. the analysts would be able to show the software actually dealing with requests that the client would typically give it in real life. A live scenario can also help analysts to see if anything in the client's environment or set-up hasn't yet been taken into consideration or would make the software ineffective. The analysts' Team Leader said to me: "We represent our software to a client by developing a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment. It really helps us to explain our software functions and connect these functions to a real case". [Team Leader]. The Team Leader also told me that using these live scenarios was usually a successful strategy. When the analysts create a live scenario of a business case, they usually succeed in showing the client that the software they have can deliver a good solution for the client. The analysts had already prepared for the software demonstration by looking at the report that had been prepared by the marketing and sales team. This report gave them information about the client's organisation structure and the initial issues that the client had reported. At this live scenario, the analysts showed the client some 'H20' software that was designed to provide the functions needed by the client's restaurant. The client has a restaurant that has two different types of menu (one for local customers and one for tourists)		

<table>
<tr><td></td><td colspan="4">and three different types of service (takeaway, delivery, and internet), so they need software that can cope with these different menus and kinds of service. The client already has some software that they use, but they have a number of issues with the software. The issues that they had reported to the sales and marketing team and said again to the analysts at this demonstration were that they had an inability to follow various food orders, they were missing orders in the kitchen, and they had difficulties with maintaining an accurate inventory of items. At the beginning of the meeting, the analyst made sure he confirmed that these were the correct issues to deal with, and asked for clarification of the issues when needed. He used the marketing and sales team's report to ask the client some questions about the issues they faced. The analyst then set the software going, making sure it was running on the right settings. He started presenting the main parts of the software and then explaining how the software would be able to solve the client's current problems. I noted that the packaged solution that the analyst demonstrated actually didn't only consist of software, but also included some hardware functionalities. For example, in addition to the software functionalities provided, there was also a personal digital assistant (PDA) that could be used to take customer orders. In addition to showing the client the PDA, the analyst listed a set of functions that the software could provide for the client: i.e. it could make it easy to take an order, make sure that the kitchen successfully received the order, help the customer ask for the correct payment required, and create a record of having been paid. A couple of times during the live software demonstration, sometimes the client pointed out that the software functionality didn't support the client's business. When this happened, the analyst took notes and agreed that he would either add the function now or modify it at a later stage. He also made sure to confirm with the client that he had a proper understanding of the example and problem that had been described that had been given by the client. For example, client wanted three different printers in different parts of the kitchen to be connected with PDA. In the case that functions would be modified, the analyst said that he would estimate the time and cost of any customisation effort and get back to the client about it. The only thing that really seemed to have a negative effect on using the live case scenario was the time constraint. The analysts only had a set period of time in which they could demonstrate the software to the client. So they had to do their best to show all of the main features of the software to the client during that time, and to make sure that they had successfully shown that the product could meet the client's needs and that they had an accurate idea of what the client's needs actually were. Since it is difficult to achieve all this within a short period of time (usually one or two hours; this time the analyst had 1 hours), the analyst made sure that during the live scenario he kept linking the software's functionalities to the particular business case.</td></tr>
<tr><td rowspan="4"><b>Immediate interpretations of events</b></td><td><b>Situation (where)</b></td><td><b>Process (when)</b></td><td><b>Practice (How)</b></td><td><b>Causal Explanations (why)</b></td></tr>
<tr><td rowspan="3">Pre-Implementation</td><td rowspan="3">Demonstration presentation</td><td>Live case
Explaining product functionality
Present the software</td><td>• To convince the client about the product's functions
• To develop a real case scenario that can simulate and cover various aspects of a real situation within the client's work environment
• To represent the capabilities of the system that could be most effectively demonstrated with a live case
• To be capable of delivering the possible solution that the client needs</td></tr>
<tr><td>Clarifying and confirming</td><td>• To get an understanding of the example given by the client</td></tr>
<tr><td>Initial Needs
Evaluation of</td><td>• To add the function or modify it at a later stage
• To estimate cost and time of customisation effort for users' needs.</td></tr>
</table>

			users' needs	
			Preparation	• To demonstrate the software functions. • Time constraint
			Marketing and sales team's report	• To provide the client's organisation structure and the initial issues

March 8, 2012 Organization 1 – Pre-implementation - Software demonstartion - Time Constraints in pre-implementation				
Events	Pre – Implementation/ Software demonstration		**Actors**	User4
Diary Observation (translation, notes)	• There are constraints on the time allowed for software demonstrations • Analysts concentrate on very specific client issues during demonstrations. • Analysts created strategies that related closely to the client's business processes, and in which they tried to minimise coverage of unnecessary information in order to help the client feel comfortable. • Of note is that the analysts focused on the business dimension more than on the software analysis dimension. • This is understandable given that the software offer had not yet been accepted by the client.			
Meeting Notes				
Descriptive analysis	Today I asked the Team Leader from the other organisation about the ways that time constraints impacted on the pre-implementation stage, but primarily on the software presentation. I was interested in this specifically since I just witnessed the H2O presentation yesterday. The Team Leader told me that because there are time constraints on software demonstrations, in their organisation they definitely do not try to provide a comprehensive explanation or demonstration of software during any software presentation. Instead, they create a demonstration in which they target discussing and dealing with very specific issues: generally the issues that the client had already raised to the marketing and sales team. So, rather than show the client everything about the software, the analysts concentrate on showing the parts of the software that would deal with any issues the client has already raised or provide the functionalities that the client wants. I was told by the Team Leader that the analysts follow demonstration strategies that relate closely to the client's business processes. Meanwhile, they try to minimise mentioning information about the software that isn't necessary at this stage, that wouldn't help them sell the software, or that would just overwhelm the client. They minimise the information presented by focusing more on the business dimension of what the software can do for the client in practical terms, rather than on actually speaking about the different aspects of the software. They feel that this is the best strategy for getting the client to accept a software offer and for making the client feel comfortable. From what the Team Leader said, a lot of the time the time constraint on software demonstrations is actually self-imposed by the software company. They keep the demonstrations short on purpose so as not to overwhelm or bore the client: "The flexibility that we want to have during software demonstration is constrained by a time limit since we only have an hour and a half to present our software … so we have to do our best to explain our software functions to the client … Our strategy is not to make the client stay for a long time in the software demonstration … we do not want to make the client feel tired and we do not want to spend a lot of time explaining software that really needs months to explain; rather, we focus on the main issue and provide a live scenario through which the software can demonstrate a possible solution". [Team Leader]. So it looks like Organisation 1 likes the idea of using live scenarios, for similar reasons to Organisation 2.			
Immediate interpretations of events	**Situation (where)**	**Process (when)**	**Practice (How)**	**Causal Explanations (why)**
	Pre-Implementation	Software demonstartion	Dealing with time constraint	• Do not try to provide a comprehensive explanation or demonstration of software during any software presentation. • Demonstration strategies that relate closely to the client's business processes • To minimise mentioning information about the software that isn't necessary at this stage, that wouldn't help them sell the software, or that would just overwhelm the client.

			Usage of sales and marketing team report	• To concentrate on showing the parts of the software that would deal with any issues the client has already raised or provide the functionalities that the client wants
			Focusing on business aspect	• Focusing more on the business dimension of what the software can do for the client in practical terms, rather than on actually speaking about the different aspects of the software. • It is a strategy for getting the client to accept a software offer and for making the client feel comfortable

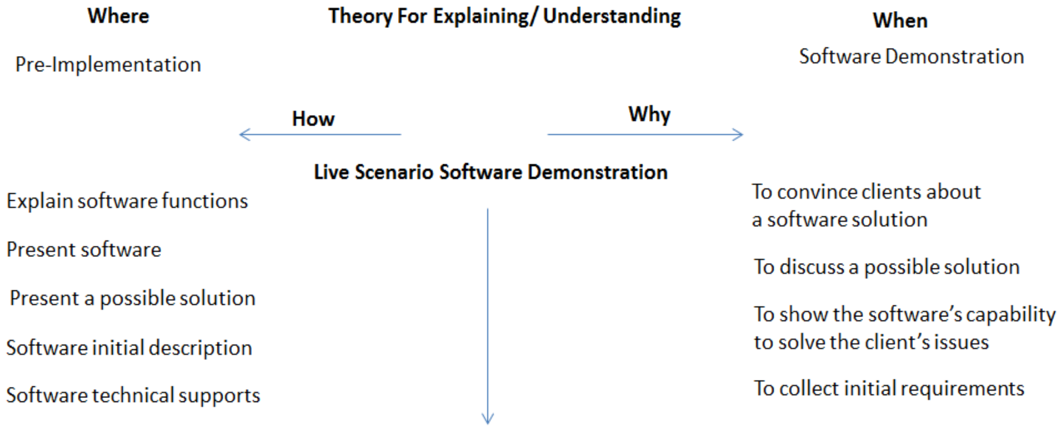
March 11, 2012 - Organization 2 - Installation				
Events	Installation		**Actors**	User5
Diary Observation (translation, notes)	• The client had accepted the software offer • Several issues were discovered by installing the copy of the software. For example, issues were found that related to server compatibility, such as speed, space, and RAM size. • Other issues were found on the client-server side, such as the client-server computer having some missing components that were related to running Dell files. • The analysts start to identify potential misalignments via the use of a copy of the packaged software. • The analysts spoke about the use of this installation of the copy as a way to educate users about the software's functionalities, to help users create a vision about the software functions, and to increase users' participation in discussions.			
Meeting Notes	**Du - Implementation**	**Organization 2/ Client Site D**		**HR**
	• Installing the copy of the software on server and some issues have been discovered • Installing the copy of the software at client-server computers. • Clients sign the installing report action.			
Descriptive analysis	Today I accompanied analysts from Organisation 2 when they installed a copy of their software at the client's site. The client has already accepted the software offer and the analysts now want to collect more information about the client's needs and the client's current infrastructure by installing a copy of the packaged software. The copy that was installed is a full copy of the software (not just a simpler 'test' copy) but might require some customisation due to issues that the analysts discovered today. From speaking to the Team Leader of the analysts about the purpose of installing the packaged software today, I learned that the analysts frequently install copies of the software in this way in order to discover any issues that relate to the client's/users' infrastructure, in order to get more information about the users' business process and requirements and the features that the client wants, and in order to better identify misalignments. This can tell them a lot about the client organisation and about what is required of the software. Installing the software in this way also gives the analysts a chance to educate the client and their users about the software product's functionalities. When I discussed this with a few different analysts, the analysts told me that installing the copy was a way to educate users about the software's functionalities, to help users create a vision of the software and how it could function for them, and to increase the users' participation in discussions. The Team Leader also mentioned that this kind of installation was useful because it was more effective than just trying to ask an organisation's IT people about their organisation's infrastructure: "We cannot ask users about their infrastructure because most of the users are not IT people and even IT people don't know some of the infrastructure requirements for software to be run … We try to discover any issues with the users' infrastructure, but we do that by installing a copy of our software … that's the only way to get to know the issues with the users' infrastructure". [Team Leader]. The main things that happened during the meeting today were that the analysts installed the software at client-server computers, and this related in their finding a range of problems related to how the software ran on the client's current infrastructure. These problems could be considered potential misalignments. Some of the issues that were discovered were that there were problems with the server compatibility: these problems related to speed, space, and RAM size. There were also problems on the client-server side. The client-server computer was missing some components that would be needed to run Dell files. The Team Leader explained to me about the problems with the client-server computer and the desktops at the client's organisation: "Everybody had to go and visit each of the desktops to install the apps … the users' computers are not compatible with our software requirements so we have to fix them". [Team Leader]. At the end of the meeting, the clients signed a report stating that the installation had taken place. It appears that the analysts make sure to document every action of this kind as a way of managing the progress of different projects.			
Immediate	**Situation (where)**	**Process (when)**	**Practice (How)**	**Causal Explanations (why)**

interpretations of events	During-Implementation	Installation, identify technical needs, and Identifying misalignments	Software explanation	• To determine what features wants and needs the user may have in relation to the software on offer • To get details of the users' business process and the requirements related to the software scope.
			Using a copy of the packaged software	• To start collecting more details about the user's needs • To educate users about the software's functionalities • To help users create a vision of the software and how it could function for them, and to increase the users' participation in discussions.
			Installing the copy of the software	• To discover any issues with the users' infrastructure
			Sign the installing report	• To manage the work's progress.

April 5, 7 2012 - Requirements Documentation Practices			
Events	Focus Group	**Actors**	User2, User3, User4, User5, User6, User7 and User8
\Descriptive analysis	We first discussed 'define a standard document structure'. The analysts understood this to mean there is a standard form that they use in order to document users' needs and their statements regarding this practice confirmed that they use this practice a great deal – in fact, analysts from each organisation told me that it was always used in their organisation. This practice therefore was found to be a 'standardised practice' in PSIRE. One analyst [user4] told me they considered it highly important because "we use to exchange information within our company and to manage our work". Other practices that they said they used all the time/every time, thus in a standardised way, were 'include a summary of the requirements', and 'make a business case for the system'. My task at this point was not so much to ask analysts why they used practices the way they did, but simply to find out to what degree they used them; however, sometimes the analysts did comment in ways that provided reasons for the level of use. For example, they always used and included 'a summary of the requirements' in requirements documentation because this seemed like such a basic necessity for paperwork and recording the documents would serve as the basis for all further development of packaged software. One analyst [user5] told me "we always make a summary of the requirements because this helps us make sure that every business need we're thinking about is linked to an actual requirement. We also want all of those requirements to be linked to a deliverable". They also always made 'a business case for the system' because it was important to clarify what users' needs are and what the changes were that they asked for regarding the packaged software functionalities. User5 said that "we use a misalignments specification form to explain a business case that users ask us to change with a specific function so developers will know where the changes should be". I found that PSIRE already had differences from traditional RE when the analysts said that not all of the practices on the list were things that they accorded standardised use. For example, most of the practices on the list for 'requirements documentation practices' only received 'common use' by them. Analysts said, for example, regarding 'explain how to use the document' that they did this quite of the misalignments specification form and software demonstration request form are very structured and the language used on the form is formal. Helping readers find information on the documents was also only given common use, not a practice used all the time. Analysts said that they sometimes did this carefully but did not bother to do it all of the time because of time constraints and because it was not that difficult to look through and find information anyway. They only did this practice for very big or complex jobs or for reports with many pages. It was therefore obviously something that they thought was helpful to do, but it wasn't essential that they do it. I found from what the analysts said that the practice 'make the document easy to change' also had common use. Most of them said they did this most of the time. They suggested that they usually did this "in case we need to change what users have asked us to do. We have to make it so any other analysts who have the permission to modify the users' request can do it in an easy way" [User3]. However, they didn't always do it because "normally only one or two of us deal with each new piece of software, so we already know what the users' needs are and we don't need any other analysts to make any changes". *What was an interesting result from these Focus Groups was that I found that* there were actually a whole lot of requirements engineering practices related to requirements documentation that the analysts practiced and could tell me about that had not been included on my list. Since they didn't match up with any of the practices on my list and therefore didn't match up with practices generally used in traditional RE, I realised that they were talking about new requirements documentation practices that were used only in PSIRE. The new practices they told me about were creating or including a 'users' needs/misalignment document', 'estimating time needed for a users' needs document', 'estimating cost needed for a users' needs document', and including a 'users' needs validation document'. It was interesting to hear about these new practices, especially to notice the fact that the analysts seemed very interested in first estimating the time and cost involved with creating users' needs documents. One analyst, User6, said that "we estimate the time and cost it could take to create a users' needs document because it helps us keep track of the resources we might need for a project. That can help us decide whether the project is worthwhile, and estimate the scope of the project before starting it". It was also interesting to note that they sought later validation of the initial user needs and would document this validation. From what they said, they thought it important to check this		

	to make sure they understood the client's needs properly and to keep a proper track of everything. User5 noted that "the validation of user needs can take place during our software demonstrations"	
Immediate interpretations of events		
Practice (How)	**Causal Explanations (why)**	**Level of Practiced**
Define a standard document structure	To document users' needs and their statements regarding this practice confirmed that they use this practice a great deal. To exchange information within the analysts' company and to manage their work	Standardised use
Explain how to use the document	Once we have a new developers or once the developers did not work on the software before so they need to know how the form link between users' needs and software functions	Common use
Include a summary of the requirements	A basic necessity for paperwork and recording the documents would serve as the basis for all further development of packaged software	Standardised use
Make a business case for the system	This ensures that each business need is linked to an actual requirement, and that each requirement is linked to a deliverable. To clarify what users' needs and what the changes they asked regarding the packaged software functionalities	Standardised use
Define specialised terms		Discretionary use
Make document layout readable	They can understand each other's writing or use of terminology anyway and how they fill out the forms and they are well-trained enough that they already understand the use of jargon used by others	Common use
Help readers find information	Time constraints/ it were not really that difficult to look through and find information anyway. Did this practice for very big or complex jobs or for reports with many pages	Common use
Make the document easy to change	"if they need to change what users have asked them to do so any other analysts who has the permission to modify the users request can do it in easy way" but that they didn't always do it because "normally one or two of us deal with each software so we know what the users' needs and it is not required from other to make any change"	Common use
Misalignments specification document	To exchange information within the analysts' company and to manage their work. To define the relation to other requirements, to other solution components, and to other artefacts.	Standardised use
Estimating cost and time needed for users' needs document	To notice the fact that they seemed very interested in first estimating the time and cost involved with creating users' needs documents. To be a way of keeping track of the possible resources needed for a project and deciding whether the project was worthwhile, and estimating the scope of the project before beginning it.	Standardised use
Include users' needs validation document	To check this to make sure they understood the client's needs properly and to keep a proper track of everything	Standardised use

Appendix G: Toward building a theory



Where

Pre-Implementation

Theory For Explaining/ Understanding

When

Software Demonstration

How

Why

Live Scenario Software Demonstration

Explain software functions

Present software

Present a possible solution

Software initial description

Software technical supports

To convince clients about a software solution

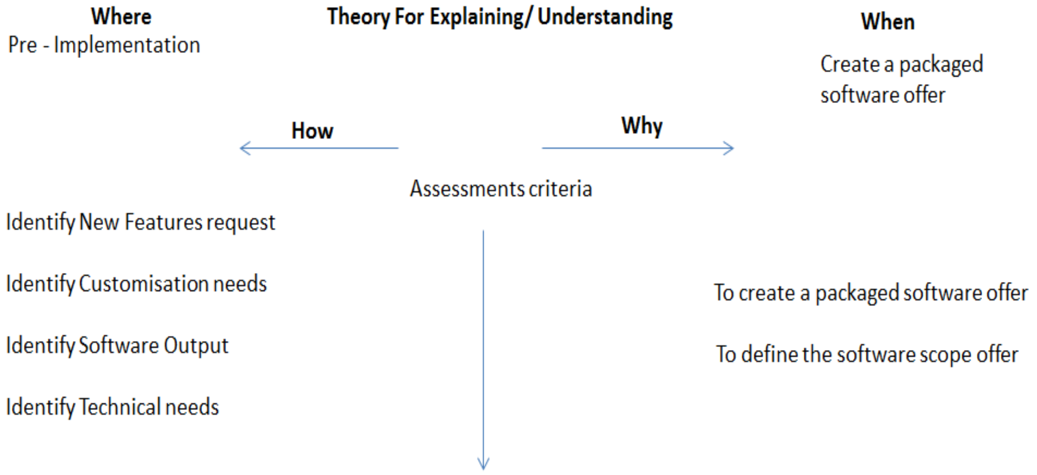
To discuss a possible solution

To show the software's capability to solve the client's issues

To collect initial requirements

Higher-level Theoretical Concept

'Live scenario software demonstration' was conceptualized as a strategy that was intended to help analysts convince clients about a software solution, as they demonstrate, and discuss, a possible solution.



Where

Pre - Implementation

Theory For Explaining/ Understanding

When

Create a packaged software offer

How

Why

Assessments criteria

Identify New Features request

Identify Customisation needs

Identify Software Output

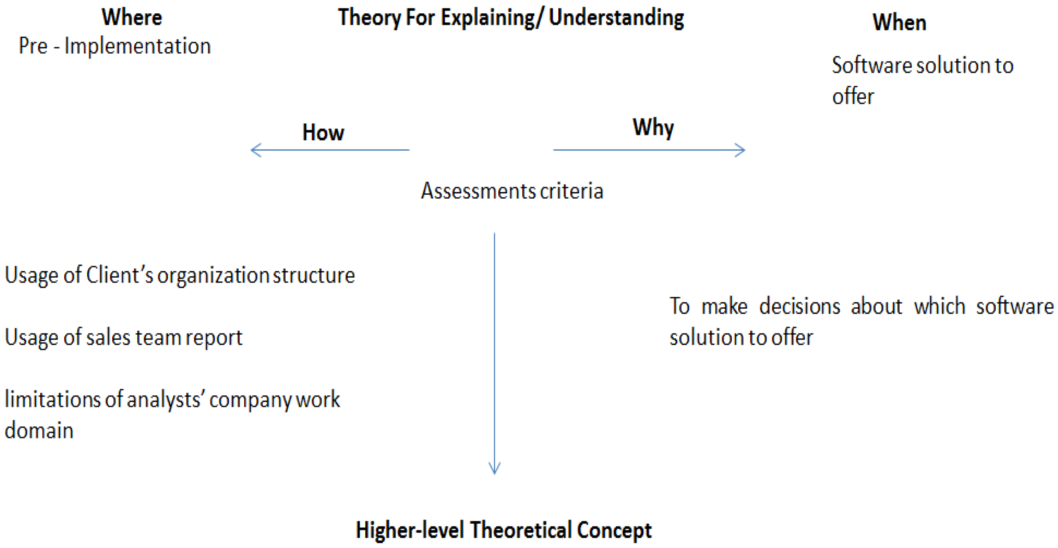
Identify Technical needs

To create a packaged software offer

To define the software scope offer

Higher-level Theoretical Concept

The New Features request, Customisation needs, Software Output, and Technical needs were conceptualized as assessments criteria that were intended to help the company to define software scope and create a packaged software offer



Where
Pre - Implementation

Theory For Explaining/ Understanding

When

Software solution to offer

How

Why

Assessments criteria

Usage of Client's organization structure

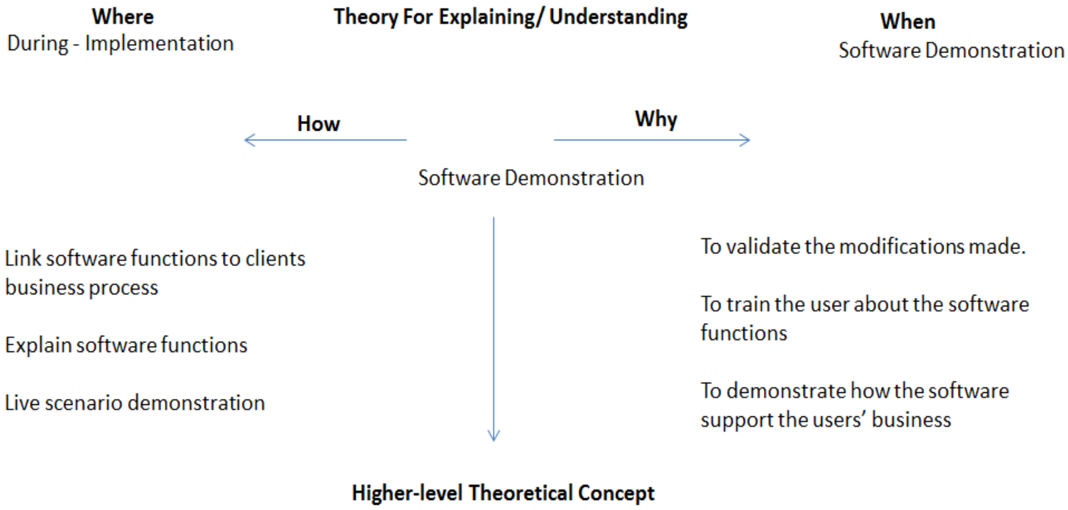
Usage of sales team report

limitations of analysts' company work domain

To make decisions about which software solution to offer

Higher-level Theoretical Concept

'Client's organization structure and the limitations of analysts' company work domain' were conceptualized as assessments criteria that were intended to help analysts to make decisions about which software solution to offer



Where
During - Implementation

Theory For Explaining/ Understanding

When
Software Demonstration

How

Why

Software Demonstration

Link software functions to clients business process

Explain software functions

Live scenario demonstration

To validate the modifications made.

To train the user about the software functions

To demonstrate how the software support the users' business

Higher-level Theoretical Concept

Software demonstration was conceptualized as a strategy that was intended to help analysts to explain software functions, train users, and validate the modifications made.

Appendix H: RE practices checklist

We are interested in understanding which practices you use in requirements engineering. For each practice shown in the following tables place a tick in the column that indicates whether you either personally used or witnessed someone else on your team use that practice during packaged software implementation process:

- Standardised use (SU): This practice has a documented standard and is always followed as part of the organisation's software development process i.e. it is mandatory.
- Common use (CU): This practice is widely followed in the organisation but is not mandatory.
- Discretionary use (DU): This practice is used at the discretion of individual project managers. Some may have introduced the practice for a particular project.
- Never used (NU): The practice is never or rarely applied.

Requirements elicitation Practices – Result validation

Requirements Elicitation Practices				
No	**Traditional RE Practices**	**Type**	**PSIRE**	**Validation**
RE1	Assess system feasibility	Basic	SU	SU
RE2	Be sensitive to organisational and political consideration	Basic	SU	CU - SU
RE3	Identify and consult system stakeholders	Basic	SU	SU - CU
RE4	Record requirements sources	Basic	SU	CU
RE5	Define the system's operating environment	Basic	SU	SU
RE6	Use business concerns to drive requirements elicitation	Basic	SU	CU - SU
RE7	Look for domain constraints	Intermediate	DU	DU
RE8	Record requirements rationale	Intermediate	CU	CU
RE9	Collect requirements from multiple viewpoints	Intermediate	***DU***	***SU***
RE10	Prototype poorly understood requirements	Intermediate	SU	SU - CU
RE11	Use scenarios to elicit requirements	Intermediate	SU	CU
RE12	Define operational processes	Intermediate	***DU***	***CU***
RE13	Reuse requirements	Advanced	SU	SU - CU

New Requirements elicitation Practices – Result validation

New Requirements Elicitation Practices				
No	**RE Practices**	**Type**	**PSIRE**	**Validation**
RE14	Use live software demonstration to elicit users' needs	Basic	SU	SU
RE15	Use a user manual	Basic	SU	SU

Requirements analysis and negotiation Practices – Result Validation

Requirements Analysis and Negotiation Practices				
No	**Traditional RE Practices**	**Type**	**PSIRE**	**Validation**
RA1	Define system boundaries	Basic	SU	SU
RA2	Use checklists for requirements analysis	Basic	DU	DU
RA3	Provide software to support negotiations	Basic	SU	CU - SU
RA4	Plan for conflicts and conflict resolution	Basic	SU	SU - CU
RA5	Prioritise requirements	Basic	***DU***	***SU - CU***
RA6	Classify requirements using a multi-dimensional approach	Intermediate	***SU***	***CU - DU***
RA7	Use interaction matrices to find conflicts and overlaps	Intermediate	DU	DU
RA8	Assess requirements risks	Advanced	SU	SU

New Requirements analysis and negotiation Practices – Result Validation

New Requirements Analysis and Negotiation Practices				
No	**RE Practices**	**Type**	**PSIRE**	**Validation**
RA9	Use print-out of a screen shot to clarify conflicts, and engaging in conflict resolution	Basic	SU	SU - CU
RA10	Use live case scenarios to support negotiations	Basic	SU	SU

Describing requirements Practices – Result Validation

Describing Requirements Practices				
No	**Traditional RE Practices**	**Type**	**PSIRE**	**Validation**
DR1	Define standard templates for describing requirements	Basic	SU	CU - SU
DR2	Use languages simply and concisely	Basic	SU	SU
DR3	Use diagrams appropriately	Basic	***DU***	***CU - SU***
DR4	Supplement natural language with other description of requirement	Basic	***SU***	***CU - DU***
DR5	Specify requirements quantitatively	Intermediate	DU	CU - DU

New Describing requirements Practices – Result Validation

New Describing Requirements Practices				
No	**RE Practices**	**Type**	**PSIRE**	**Validation**
DR6	Specify relationship between users' needs and other software functions	Basic	SU	SU
DR7	Specify relationship between users' needs and data stores	Basic	SU	SU

System modelling Practices – Result Validation

System Modelling Practices				
No	**Traditional RE Practices**	**Type**	**PSIRE**	**Validation**
SM1	Develop complementary system models	Basic	NU	DU - NU
SM2	Model the system's environment	Basic	NU	NU
SM3	Model the system architecture	Basic	NU	NU
SM4	Use structured methods for system modelling	Intermediate	***DU***	***SU - CU***
SM5	Use a data dictionary	Intermediate	CU	CU - DU
SM6	Document the links between stakeholder requirements and system models	Intermediate	SU	CU

Requirements validation Practices – Result Validation

Requirements Validation Practices				
No	**Traditional RE Practices**	**Type**	**PSIRE**	**Validation**
RV1	Check that the requirements document meets your standards	Basic	SU	SU
RV2	Organise formal requirements inspections	Basic	DU	CU - DU
RV3	Use multi-disciplinary teams to review requirements	Basic	DU	CU - DU
RV4	Define validation checklists	Basic	CU	CU - SU
RV5	Use prototyping to animate requirements	Intermediate	SU	SU - CU
RV6	Use a draft user manual	Intermediate	SU	SU
RV7	Propose requirements test cases	Intermediate	SU	SU
RV8	Paraphrase system models	Advanced	***DU***	***CU***

Requirements management Practices – Result Validation

Requirements Management Practices				
No	**Traditional RE Practices**	**Type**	**PSIRE**	**Validation**
RM1	Uniquely identify each requirement	Basic	CU	SU - CU
RM2	Define policies for requirements management inspections	Basic	SU	SU
RM3	Define traceability policies	Basic	DU	CU - DU
RM4	Maintain a traceability manual	Basic	DU	DU
RM5	Use a database to manage requirements	Intermediate	SU	SU
RM6	Define change management policies	Intermediate	SU	CU
RM7	Identify global system requirements	Intermediate	***DU***	***CU***
RM8	Identify volatile requirements	Advanced	DU	DU - CU
RM9	Record rejected requirements	Advanced	SU	CU - SU

Appendix I: Schedule of the what, when, and where

1 February 2012
Wed

All Day Pre – Implementation/ Software demonstration/ Inventory and purchases -- Organization 1

5 February 2012
Sun

All Day Pre – Implementation/ Software demonstration/ Accounting Software -- Organization 1

8 February 2012
Wed

All Day Pre – Implementation/ Software demonstration/ HR -- Organization 2/ Client site

9 February 2012
Thu

All Day Software Offer -- Organization 2

11 February 2012
Sat

All Day PSI scope, PSI offer, and PS demonstration -- FG: Organization 1

12 February 2012
Sun

All Day Installation -- Organization 1/ Client Site

14 February 2012
Tue

All Day Misalignments Specification Form -- Organization 2

16 February 2012
Thu

All Day PSI scope, PSI offer, and PS demonstration -- FG: Organization 2

21 February 2012
Tue

All Day Software offer -- Organization 1

23 February 2012
Thu

All Day PS demonstration -- FG: Organization 1

25 February 2012
Sat

All Day PS demonstration -- FG: Organization 2

26 February 2012
Sun

All Day Software demonstration, identify users’ needs, validation of users’ needs / HR -- Organization 2/ Client site

27 February 2012
Mon

All Day Respond to the discovery of a misaligments -- Organization 2

7 March 2012
Wed

All Day Pre – Implementation/ Software demonstration/ H2O -- Organization 2

8 March 2012
Thu

All Day Pre-implementation - Software demonstartion - Time Constraints in pre-implementation -- Organization 1

10 March 2012
Sat

All Day PS customisation. -- FG: Organization 1

11 March 2012
Sun

All Day Installation -- Organization 2/ Client Site D

12 March 2012
Mon

All Day During Implementation – Software demonstration & identify users’ needs
-- Organization 1/ Client
Site B

13 March 2012
Tue

All Day During Implementation – Software demonstration & identify users’ needs
-- Organization 1/ Client
Site A

15 March 2012
Thu

All Day Pre – Implementation/ Software demonstration -- Organization 1

17 March 2012
Sat

All Day PS customisation. -- FG: Organization 2

21 March 2012
Wed

All Day During Implementation – Software demonstration & validation of
users’ needs -- Organization 1/ Client Site A Modifications done

22 March 2012
Thu

All Day PS customisation, Identify misfits -- FG: Organization 1

24 March 2012
Sat

All Day PS customisation, Identify misfits -- FG: Organization 1

26 March 2012
Mon

All Day	During Implementation – Software demonstration & validation of users' needs -- Organization 1/ Client Site B

27 March 2012 Tue

All Day	Installation -- Organization 2/ Client C

29 March 2012 Thu

All Day	PS customisation, Identify misfits -- FG: Organization 2