GROUPWARE REQUIREMENTS
EVOLUTION PATTERNS

Dulce Trinidad Pumareja
Ph.D. Dissertation committee:

Chairman and secretary
Prof. dr. ir. A.J. Moutaanan  University of Twente, The Netherlands

Promotors
Prof. dr. R.J. Wieringa  University of Twente, The Netherlands
Prof. dr. J.C. Looise  University of Twente, The Netherlands

Assistant promotors
Dr. N. Sikkel  University of Twente, The Netherlands

Members
Prof. Dr. D. M. Berry  University of Waterloo, Canada
Prof. dr. B. Paech  University of Heidelberg, Germany
Prof. dr. V. Evers  University of Twente, The Netherlands
Prof. dr. N.E.J. Oudshoorn  University of Twente, The Netherlands
Dr. ir. M.J. van Riemsdijk  University of Twente, The Netherlands

CTIT Ph.D. Thesis Series No. 12-235
Centre for Telematics and Information Technology
P.O. Box 217, 7500 AE
Enschede, The Netherlands

SIKS Dissertation Series No. 2013-005
The research reported in this thesis has been carried out under the auspices of SIKS, the
Dutch Research School for Information and Knowledge Systems.

Printed and bound by: Print Service Ede
Cover image and design: Renée Bloupot-Pumareja

ISSN: 1381-3617
DOI: 10.3990/1.978903653061

Copyright © 2013 Dulce Trinidad Pumareja

All rights reserved. No part of this book may be reproduced or transmitted in any form of by any means, electronic or mechanical, including photography, recording or any information storage and retrieval system, without prior written permission of the author.
GROUPWARE REQUIREMENTS
EVOLUTION PATTERNS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
Prof. dr. H. Brinksma,
on the account of the decision of the graduation committee,
to be publicly defended
on Thursday, January 17th, 2013 at 14.45 hrs.

by

Dulce Trinidad Pumareja

born on November 10, 1971
in Quezon City, Philippines
This dissertation is approved by:

Prof. dr. R.J. Wieringa (promotor)
Prof.dr. J.C. Looise (promotor)
ABSTRACT

Requirements evolution is a generally known problem in software development. Requirements are known to change all throughout a system's lifecycle. Nevertheless, requirements evolution is a poorly understood phenomenon. Most studies on requirements evolution focus on changes to written specifications and on software architecture and design. Usually, the focus is when the software is under development. Little is known about how requirements evolve when software is put into use.

Groupware is an example of an application domain in which the requirements continue to evolve after the system is deployed to the organization. Groupware is any ICT (software + hardware) application that supports the cooperative processes of individuals working as a group. Increasingly, groupware functionality is becoming more present in today's business applications and large information systems. The cooperative processes supported by a groupware application have no structure. Rather, its structure evolves in a way that cannot be specified in advance and arises spontaneously. Therefore, how a groupware system will be used in its operating environment cannot be anticipated in advance. There is also the added complication that groupware requirements are difficult to elicit due to the elusive nature of cooperative work. As software for supporting the cooperative processes of people working together, groupware technology has the potential to bring about profound organizational changes. Various studies of groupware implementation point to emergent organizational properties. The interaction between users and software leads to improvements in performance, new forms of communication, changes to group structure and functioning, all of which indicate that requirements have changed.

This study is an empirical investigation of requirements evolution for groupware systems in use by means of case studies. Its goal is to contribute to the development of a theory of requirements evolution. A conceptual framework offering an integrated view of requirements as a collection of domains was developed to guide and structure the investigation. The view takes the broad dimensions of business, software, problems, and solutions as requirements thus giving rise to four domains of requirements: business problem, business solution, software product concept, and software solution specification. Requirements evolution is initially formulated as the change in requirements over the course of time.

The application domain of groupware was chosen as the empirical setting in which to observe requirements evolution during system use. Four case studies of groupware implementations were conducted. Two failed implementations and two successful implementations were investigated. The conceptual framework is used to analyze the cases and is updated and improved based on an evaluation of how useful has it been in providing insights about requirements evolution. A final version of the framework is developed and this is used to analyze the last two case studies.

The results include the discovery of impact relations: commonly recurring mechanisms by which changed and new requirements lead to other requirements in different areas. Ultimately, requirements evolution is the resolution of a breakdown or an initiative resulting in an impact relation. The most important contribution of this research is a set of requirements evolution patterns: aggregations of impact relation sequences that explain the mechanisms underlying awkwardly familiar patterns of behavior in system implementation.
ACKNOWLEDGEMENTS

There are not enough words to express my gratitude and heartfelt thanks to all those who have helped realize this work.

My sincere thanks to my promotor Roel Wieringa who continued to believe and motivate the completion of this thesis. More importantly, I really appreciate your support for a theory development approach to requirements engineering research and the choice to move forward by means of a case study methodology.

I also would like to thank my co-promotor Klaas Sikkel who had been more than a daily supervisor and mentor for me. Thank you Klaas for the discussions, for the time you spent giving feedback and for following up especially when coming to the faculty became less frequent due to work.

To co-promotor Jan Kees Looise: your inputs and perspectives representing the non-computer science domain is well appreciated.

To the members of the dissertation committee: thank you very much for your time and patience in reading, reviewing and providing feedback to the manuscript. To Dan Berry: thank you for being pro-active in the process and for including English language corrections in your feedback.

Without data, there would be no results to report. Without the contribution of the respondents in the case studies, there would be no empirical basis for analysis, for improving the conceptual framework and for discovering change patterns. Therefore, this research owes it to each individual respondent in the case studies: the project manager, the actuaries and product managers at the insurance company in the first case study; the teachers at the Seminarium voor Orthopedagogiek in the second case study; the teachers and students at the Faculty of Industrial Design Engineering in the third case study and my former colleagues based in Friesland in the last case study. Specifically, I would like to thank Tanya Bondarouk for her company and collaboration while we did the first case study together.

Once more, I would like to thank Roel for the encouragement and support given for learning the Dutch language. The benefits of such skill go beyond the pragmatics of doing case studies for research. For this, many many thanks go to Mieke Poelman for being a wonderful Dutch language teacher. Er is niets prettiger om Nederlands te leren dan bij jullie thuis met een kopje thee en een koekje.

To the administrative staff of the Information Systems Group, especially Suse: your help in going through the procedures and paperwork has been invaluable. You are a lifesaver.

A special thanks also goes to kuya William Kuipers for the Dutch translation of the abstract.

Looking back, the completion of this research has been quite a journey with its highs and lows. I
am grateful to have been able to share this experience with fellow PhD students and friends. To my former colleagues -- Zlatko, Damiano, Wouter, Xiaomeng, Lianne, Andreas, Manfred, Novica & Jelena, and friends -- Nisa Fachry and Katherine Golden: thank you for your company and friendship which made this journey quite less lonely.

To my family in the Philippines, especially my mother and my brother. I thank you for supporting me in the best way you can from where you are. To my in-laws Johan and Marga: thank you for the encouragement and your interest in what I do.

And finally, last but never the least, everdearest Renée, my spouse: you have been an enormous source of inspiration. In you I draw the strength to go on and to never give up. Thank you for being there. I look forward to spending more time together now that this milestone has been reached.

Eindhoven, December 2012
# TABLE OF CONTENTS

1 Introduction .................................................................................................................. 1  
  1.1 What this thesis is about ......................................................................................... 2  
  1.2 Research motivation ............................................................................................ 2  
  1.3 Research problem .................................................................................................. 7  
  1.4 Research strategy and methodological plan .......................................................... 8  
  1.5 Structure of the thesis ........................................................................................... 9  

2 Theoretical Basis and Conceptual Framework ............................................................ 11  
  2.1 Requirements – what ............................................................................................ 11  
  2.2 Requirements process - how .................................................................................. 16  
  2.3 Requirements evolution ........................................................................................ 20  
  2.4 Groupware: an overview ....................................................................................... 23  
  2.5 Requirements evolution in groupware application domains ................................ 28  
  2.6 Towards a conceptual framework for requirements evolution ............................ 33  
  2.7 Requirements evolution conceptual framework: initial formulation .................. 34  
  2.8 Next steps: operationalization and research design ............................................. 36  

3 Methodological Justification ....................................................................................... 39  
  3.1 Case study strategy ............................................................................................. 39  
  3.2 Case study as our chosen research strategy ......................................................... 41  
  3.3 Case study design ................................................................................................ 42  
  3.4 Data analysis protocol .......................................................................................... 49  
  3.5 Challenges in data collection ................................................................................ 50  
  3.6 Case write-up ....................................................................................................... 51  

4 Case Study A: KennisNet@ Active Insurance Group .................................................. 53  
  4.1 Case background ................................................................................................... 53  
  4.2 Requirements at pre-implementation: from strategy to technology .................... 56  
  4.3 Requirements at early implementation: involving end-users in system design .... 59  
  4.4 Requirements at post-deployment: removal or re-orientation to another use? .... 63  
  4.5 Requirements evolution as within-domain change ............................................. 67  
  4.6 Requirements evolution as inter-domain change: impact relations .................... 69  
  4.7 Discussion ........................................................................................................... 76  

5 Case Study B: First Class Experience @ Central City College ..................................... 81  
  5.1 Case background .................................................................................................. 81  
  5.2 Requirements at pre-implementation: fortuitous application discovery ............. 85  
  5.3 Requirements at early implementation: shifts in implementation strategy ........ 87  
  5.4 Requirements at post deployment: from groupware to eduware ......................... 92  
  5.5 Requirements evolution as within-domain change ............................................. 97  
  5.6 Requirements evolution: impact relations ............................................................ 100  
  5.7 Discussion ........................................................................................................... 103  

6 A Conceptual Framework of Requirements Evolution .............................................. 109  
  6.1 Requirements evolution as within-domain evolution ......................................... 109  
  6.2 Requirements evolution as impact relations ....................................................... 110
6.3 Breakdowns and initiatives...............................................................................................................................................110
6.4 Additional case studies..........................................................................................................................................................111

7 Case Study C: TeleTOP @ The Department of Industrial Design Engineering..............................................................113
7.1 Case background......................................................................................................................................................................113
7.2 Requirements at Pre-Implementation: following suit........................................................................................................116
7.3 Requirements at early implementation: propagating TeleTOP use and the fire incident.............................................119
7.4 Requirements at Post-Deployment: from a glider to a Boeing 747....................................................................................124
7.5 Requirements evolution: within-domain evolution............................................................................................................127
7.6 Impact relations.......................................................................................................................................................................131
7.7 Discussion.................................................................................................................................................................................134

8 Case Study D: Oracle iLearning @ FDS Academy..................................................................................................................137
8.1 Case background.......................................................................................................................................................................137
8.2 Requirements at pre-implementation: “we want training”..................................................................................................139
8.3 Requirements at early implementation: capacity building and expansion ......................................................................142
8.4 Requirements at post-deployment: buy out and the financial crisis..................................................................................146
8.5 Within-domain evolution ...........................................................................................................................................................151
8.6 Impact relations.......................................................................................................................................................................161
8.7 Discussion.................................................................................................................................................................................168

9 Multiple Case Analysis..............................................................................................................................................................171
9.1 Case studies overview..............................................................................................................................................................171
9.2 Multiple case analysis protocol.................................................................................................................................................172
9.3 Requirements domains...............................................................................................................................................................172
9.4 Requirements evolution: within-domain evolution.............................................................................................................174
9.5 Requirements evolution: impact relations across domains...............................................................................................178
9.6 Impact relation patterns.............................................................................................................................................................183
9.7 Mechanics of evolution: breakdowns and initiatives.........................................................................................................188
9.8 Validity issues: threats to validity.........................................................................................................................................190
9.9 Conclusion.................................................................................................................................................................................193

10 Final Remarks..............................................................................................................................................................................203
10.1 Review of research goals, questions and contribution.........................................................................................................203
10.2 Research product: conceptual framework.............................................................................................................................205
10.3 Evaluation: usefulness criteria..................................................................................................................................................207
10.4 Research hypotheses.................................................................................................................................................................208
10.5 Practical implications and recommendations.......................................................................................................................209
10.6 Directions for future work..........................................................................................................................................................212
User-Defined Index

APPENDIX A. Data Gathering Questions and Checklists.........................................................215
APPENDIX B. List of Requirements Statements.......................................................................217
APPENDIX C. M1 Impact Relation Instances...........................................................................226
APPENDIX D. M2 Impact Relation Instances...........................................................................229
APPENDIX E. M3 Impact Relation Instances...........................................................................231
APPENDIX F. M4 Impact Relation Instances...........................................................................234
APPENDIX G. M5 Impact Relation Instances...........................................................................236
APPENDIX H. M6 Impact Relation Instances...........................................................................239
APPENDIX I. M7 Impact Relation Instances...........................................................................241
BIBLIOGRAPHY.........................................................................................................................243
CHAPTER 1

INTRODUCTION

In 2003, a major disaster in the domain of high technology took place: the space shuttle Columbia broke apart into pieces upon re-entering the earth's atmosphere killing all of the 7 astronauts on board (CNN 2003; CAIB 2003). According to the accident report, the immediate physical cause of the shuttle's disintegration was overheating. This was due to a damaged thermal protection system located in the left wing of the orbiter, which was hit when a piece of insulation foam fell from the external tank during take-off. However, days before the landing, engineers at NASA had already known that there was no way the shuttle and its crew can survive if the thermal protection system had at all been breached. Fears of an impending disaster were disclosed through informal emails between NASA engineers. A series of worst-case scenarios and analyses were conducted to assess the possible impacts of the foam that hit the shuttle. The results pointed to an accident waiting to happen. One of the engineers expressed his concerns in an email:

I am admittedly erring way on the side of absolute worst-case scenarios and I don't really believe things are as bad as I'm getting ready to make them out. But I certainly believe that to not be ready for a gut-wrenching decision after seeing instrumentation in the wheel well not be there after entry is irresponsible.

Further exploring potential outcomes, this engineer describes one fatal scenario:

If belly landing is unacceptable, ditching/bailout might be next on the list. Not a good day.

Apparently, this knowledge and assessment of an impending danger never reached NASA top management. Likewise, the Columbia crew members and the engineers at the Mission Control were given no indication that the mission was in trouble (CNN 2003; CAIB 2003). Therefore when the shuttle returned to earth on 1 February 2003, no alarm was issued despite receiving readings of increased temperature. Not until all data from the sensors and contact with Columbia crew was lost did it become apparent that a crisis was at hand. A few moments later, space shuttle Columbia had disintegrated.

The Columbia Accident Investigation Board (CAIB), the body tasked to conduct a thorough investigation of the incident, faults the organizational structure within NASA as the underlying reason to the disaster (CAIB 2003). In its technical report, it states the following:

The organizational causes of this accident are rooted in the Space Shuttle Program's history and culture, including the original compromises that were required to gain approval for the Shuttle, subsequent years of resource constraints, fluctuating priorities, schedule pressures, mischaracterization of the Shuttle as operational rather than developmental, and lack of an agreed national vision for human space flight. Cultural traits and organizational practices detrimental to safety were allowed to develop, including: reliance on past success as a substitute for sound engineering
practices (such as testing to understand why systems were not performing in accordance with requirements); organizational barriers that prevented effective communication of critical safety information and stilled professional differences of opinion; lack of integrated management across program elements; and the evolution of an informal chain of command and decision-making processes that operated outside the organization’s rules (CAIB 2003, p. 9)

One specific issue cited in the report refers to requirements evolution. Specifically, the report examined the way how NASA's requirements with regards to foam shedding during take-off had evolved. In the original design requirement of the aircraft, the external tank is not supposed to shed any debris and the orbiter was also not supposed to receive debris beyond a trivial amount. However, over the course of 113 successful missions during which foam-shedding and other debris impacts continued to persist, deviation from the original requirements eventually came to be regarded as a maintenance issue and less as hazard to the vehicle and crew (CAIB 2003, p. 130). In other words, critical requirements had evolved to become maintenance concerns and NASA did not follow its own design rules. The requirements on foam-shedding were overlooked, a fatal mistake that cost the shuttle's last mission and the lives of its crew.

1.1 What this thesis is about

In the given account of the Columbia space shuttle disaster, we try to portray an image of the problem associated with requirements evolution: a phenomenon of interest this thesis aims to shed light on. Specifically, this thesis is about investigating requirements evolution in groupware application domains. By groupware application domains, we mean social settings of people having the features of a group working together. They perform cooperative processes that are amenable to support by ICT-based mediation technologies such as groupware. These settings can be found in organizations in the form of teams, workgroups or an educational unit consisting of teachers and students, among others. The NASA example illustrates the nature and dynamics of such settings.

To go about with the investigation, we have formulated a conceptual framework to describe the event by means of several analytical constructs. This framework is presented, validated and updated in this thesis. Accordingly, the application domain of groupware technologies was used as a platform for gathering observations. The validation takes the form of case studies of groupware technology use in organizational settings. By providing a language to describe the event and to account for such conceptual framework, through its analytical constructs and descriptions of what took place which trigger the identification of recognizable mechanisms, can foster an understanding of requirements evolution. Such understanding can be beneficial in structuring and managing a groupware implementation project, and for software maintenance in the post-deployment phase of a software system.

Section 1.2 of this introductory chapter provides the motivation for this research. Section 1.3 is a specification of the research problem and Section 1.4 is a preliminary list of definitions of concepts used throughout the thesis. Section 1.5 describes the research approach, and Section 1.6 details the outline of the thesis.

1.2 Research motivation

While the image of the problem given above is in the domain of critical systems, it does provide a representation of the knowledge problem (Wieringa & Heerkens 2006) this thesis addresses. First, the example provides an indication of the general problem associated with requirements, or the demands ascribed to a certain design product – namely, that the product does not satisfy the requirements and the requirement continue to evolve during the use of the product. Particularly with software products, the problem of unmet requirements and evolving requirements are known
sources of difficulties and drawbacks (Lauesen 2002; Gause & Weinberg 1989; Ewusi-Mensah 2003; Ewusi-Mensah & Przasnyski 1994). Secondly, the example reveals much about the complexities and uncertainties concerning the dynamic aspects of cooperative work in organizational settings which is instantiated by the example as a breakdown in communication and knowledge diffusion.

Taken into the domain of software, the underlying theoretical and practical problem implied by the example can be further described in terms of the following research issues. These provide the conceptual background motivating this research.

1.2.1 Crisis in application development

A software development project is more likely to fail than to succeed. Less than 30% of application development projects in industry are considered to be successful – delivered on time, on budget and with required features and functions. This is the conclusion offered by Standish Group’s (1994-2009) survey of software projects conducted for over a period of 12 years. About quarter of those projects were outright failures with almost the half (44%) considered as ‘challenged’ (Standish Group 2009). These are projects that are abandoned or cancelled before completion and were otherwise pushed for completion but with cost overruns, over the time estimates and with fewer functions and features than previously specified. The common reasons why projects fail are known to be non-technical in nature (Hull et al. 2005). Table 1-1 and 1-2 are much referred to list of common project failure and success factors published more than a decade ago by the same research group based on its survey of more than 30,000 IT projects (Standish Group 1995; 2005). The list shows that in 1994 project failures are mainly due to requirements defects: incomplete, lacking in user input, unrealistic, unnecessary and constantly changing.

*Table 1-1: Factors leading to software project failure*

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete requirements</td>
<td>13.1%</td>
</tr>
<tr>
<td>Lack of user involvement</td>
<td>12.4%</td>
</tr>
<tr>
<td>Lack of resources</td>
<td>10.6%</td>
</tr>
<tr>
<td>Unrealistic expectations</td>
<td>9.9%</td>
</tr>
<tr>
<td>Lack of executive support</td>
<td>9.3%</td>
</tr>
<tr>
<td>Changing requirements / specification</td>
<td>8.7%</td>
</tr>
<tr>
<td>Lack of planning</td>
<td>8.1%</td>
</tr>
<tr>
<td>Didn’t need it any longer</td>
<td>7.5%</td>
</tr>
<tr>
<td>Lack of IT management</td>
<td>6.2%</td>
</tr>
<tr>
<td>Technology illiteracy</td>
<td>4.3%</td>
</tr>
</tbody>
</table>

*Source: Standish Group (1994)*

*Table 1-2. Contributing factors to success*

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User involvement</td>
<td>1</td>
</tr>
<tr>
<td>Executive management support</td>
<td>2</td>
</tr>
<tr>
<td>Clear business objectives</td>
<td>3</td>
</tr>
<tr>
<td>Optimizing scope and requirements</td>
<td>4</td>
</tr>
<tr>
<td>Experience project manager</td>
<td>5</td>
</tr>
<tr>
<td>Iterative and agile process</td>
<td>6</td>
</tr>
<tr>
<td>Financial management</td>
<td>7</td>
</tr>
<tr>
<td>Skilled resources</td>
<td>8</td>
</tr>
<tr>
<td>Formal methodology</td>
<td>9</td>
</tr>
<tr>
<td>Standard tools and infrastructure</td>
<td>10</td>
</tr>
</tbody>
</table>

*Source: Standish Group (2005)*
1.2.2 Implementation failure

Software failure extends beyond the inability of software developers to produce a working or functioning system for the users. It is also the inability of the implemented software to meet the expectation of users (Ewusi-Mensah 2003). Organizational deployment of software systems with unfavorable outcomes has been well-documented in information systems research. Brynjolfsson (1993) for example investigated the productivity paradox of software and showed that studies across sectors in the US suggest minimal contribution of IT investments to economic productivity. Citing Roach (1991), it was found out that the level of IT capital per information worker has begun approaching that of the production capital per production worker. There is also the time lag in benefit pay-offs in which the impacts of new technology are not immediately felt. Time and experience are needed for a complex and novel conceptual product such as software to get used to. It takes about 2-3 years to feel the strongest organizational impact of software and 5 years for the investment to pay-off (Brynjolfsson 1993; 1992).

Multiple cases of ICT implementations have been reported as failures or challenged because users resist and reject the software (Hirschheim & Newman 1988; Orlikowski 1992; Grudin & Palen 1995; Markus 1987; Rogers 2003). The introduction of a new system is a form of change which reverse reactions can be expected. Hirschheim and Newman (1988) tell about US postal workers pouring honey and inserting paper clips in their data devices. Feeling threatened, a civil servant started storing macros locally, outside the groupware application, thereby causing breakdowns in collaboration and risks of virus infection (Pike & Wulf 1999). Aside from the underlying reasons in the given examples, mismatch between specific features of system design and characteristics of the existing organization, including elements of organizational structure has been invoked as cause for user resistance. Hirschheim and Newman (1988) refer to this as organizational invalidity – the software does not fit the individuals’ and group’s work patterns including the structure of reporting relationships among them.

There is also the phenomenon of ‘drift’, in which the software implementation in the organization takes a different course of action other than planned (Ciborra 1996; Ciborra & Andreu 2001). The implementing organization encounters unexpected circumstances that show the incompleteness and possible failure of an initial technological design without organizations having yet feasible alternatives (van Baalen & van Fenema 2005). In groupware applications, Ciborra (1996) wrote about the changing role assigned to software based on prevailing trends, i.e. groupware as knowledge management software. Quite often, implementation projects start with ambitious goals, unrealistic goals and too optimistic time estimates (Ewusi-Mensah 2003).
Therefore, when it comes down to deployment, it is either the time to adjust the goals, shift focus and to think of ways of the software can be put into more productive use. Implementation then becomes a moment of correction, adjustment and making compromises about the software that could lead to re-organization (i.e. re-assigning the project or firing the project manager), having lesser functions and focusing development on the ones that are really needed, extending deadlines, contracting training services and hiring external consultants.

1.2.3 Continuously evolving requirements

Continuously evolving requirements are central problems in software development and use (Table 1-1; Ewusi-Mensah 2003; Lauesen 2002; Pressman & Ince 2000, Gause & Weinberg 1989). In the previously given examples, requirements evolution is known to have direct and indirect impacts to the outcome of application development and implementation. Investigations of failed projects show that projects often start with very ambitious goals and later in the process right in the middle of the development, changes take place, therefore delaying the project (Ewusi-Mensah 2003; Ewusi-Mensah & Przasnyski 1994; Ewusi-Mensah 1997). In the FoxMeyer Delta project example, the pharmaceutical company embarked on an aggressive business campaign to recoup lost revenues while the project was already on its way. The addition of a new business partner with a completely different business operations model had to be integrated in the ERP system being developed. This changed the focus of the project drastically and added difficulties to the project leading to its abandonment and bankruptcy for FoxMeyer (Jesitus 1997). With regards to the issues of user resistance and software drift in software implementation, the fact that these arise indicate that requirements have changed – from application acquisition and roll-out.

Changing requirements should be considered the norm in systems development rather than having stable or fixed requirements (Ewusi-Mensah 2003). Requirements change and shift from the moment they are conceived up to the time they are formulated and specified as desired software program functionality and properties. Software developers encounter the problem of ‘requirements or feature creep’, which are late or last minute modifications to a software system resulting into additional functionality not thought of or not addressed previously. Usually, these modifications increase project cost (Boehm 1981). The study by Lutz & Mikulski (2003) shows how requirements continually evolve and get discovered even at the later phases of a system’s development such as testing and operations. Incomplete requirements as well as those poorly understood ones manifest themselves as testing and operational anomalies in a safety-critical system.

1.2.4 Limited and poor understanding of requirements evolution

Requirements evolution is a poorly understood phenomenon. The current practice in requirements engineering (Robertson & Robertson 1999; Carter et al. 2001; Leffingwell & Widrig 2003) tends to focus on process-oriented approaches aimed at managing and supporting the evolution process. With these approaches, it is possible to know that requirements have changed. What kind of change on the requirements took place, i.e. addition, deletion, modification, can also be known and described (Anderson & Felici 2002). The landscape of research into requirements evolution can be classified into two dominant research themes. These are researches covering:

• approaches and methods dealing with how to manage, control and mitigate the impact of change and evolution (Barber et al 2002; Carter et al 2001).

• Observations and descriptions of the phenomenon as well as means for modeling such (Lubars et al. 1993; Sommerville 1995, 2001; Dobson et al. 1994). Descriptions are given in terms of categories and taxonomies of requirements and the types of changes.

However, these approaches have little to say about why requirements have changed and how
these have evolved. Through the framework developed in this thesis, it is our goal to expand this body of research to explain as to why requirements change, not by describing causes of change but by identifying and describing plausible reasons for change. In addition, the focus of these approaches also tends to be more on the solution side of requirements (Hull et al. 2005) where change is characterized by a re-specification of an already existing requirement. It assumes a stable set of requirements with changing solution sets. On the other hand, there is also the problem side of requirements. This is the problem domain in which the system is going to be used (Hull et al. 2005). Requirements evolution coming from an evolving problem domain poses the issue of requirements uncertainty (Harker et al. 1993; Dobson et al. 1994). An evolving problem domain complicates the requirements capture process and makes the specification an elusive task. Our framework takes into consideration the relevance of both the problem and solution side of requirements.

The introduction of a new system to the problem domain changes the problem domain. As a system is introduced for use in an organization, the demands on this system also change. The evolution is a result of a socio-technical interaction in which system and context co-evolve with each other. From an organizational theory perspective, the evolution of the problem domain as a result of system implementation has been the focus considerable IS research (Orlikowski & Hofman 1997; Karsten & Jones 1998; Orlikowski 1996; Markus & Robey 1988). However, the implications of these to software requirements need further theoretical attention.

1.2.5 Software evolution

Systems need to continually adapt to their operating environment in order to continually remain in satisfactory use (Lehman et al. 2000). There is hardly any software currently in use today – COTS, open-source, personal productivity tools, games as well as business software that is not continually being upgraded. Across application domains especially on large commercial software, a pattern of evolutionary behavior has been observed and documented into laws of software evolution (Lehman & Belady 1985; Lehman & Ramil 2003). It is further argued that software evolution is a broader phenomenon than software maintenance. It also includes all other activities intended to maintain stakeholders' satisfaction over time. Given that requirements give rise to software, as requirements evolve, it is but natural to expect software to also evolve.

1.2.6 Requirements evolution in groupware application domains

In 1.2.2 as well as 1.2.4, we briefly touched on the issue of requirements evolution in groupware domains implied in software implementation failures. Groupware is an example of an application domain in which requirements continue to evolve after the system is introduced and used within an organization. Groupware systems are applications used by groups of people for a shared goal. Groupware technology is cooperative technology. For example, people could use shared workspaces for the goal of sharing knowledge, they use email with the goal (one hopes) of getting work done, and they use videoconferencing with the shared goal of conducting a meeting. For groupware systems, there is the additional complication that people are generally not able to state their requirements before using the system. Even observations of current work or analysis of documented work procedures are not sufficient to elicit requirements for groupware. When a groupware system is introduced in a work context, it brings about changes in that context which invalidate requirements gathered earlier (Ciborra 1996; Jarke & Pohl 1994; Orlikowski 1996; DeSanctis & Poole 1990). This is requirements uncertainty (Harker et al. 1993). Groupware requirements are inherently uncertain: work done using groupware is usually not documented in procedures, people doing the work are often not able to describe it explicitly and when groupware is introduced, it changes this work. The motivation for this research as the above issues imply is the interest in understanding the dynamics of requirements evolution, a critical factor that
influences a project's success or failure. We have chosen the application domain of groupware as the focus of the study because it represents a research area in which requirements inevitably evolve. Specifically, the evolution takes during the use of the software which means that the investigation will focus on the post deployment implementation and use phase of system implementation and use in the organization.

1.3 Research problem

This section provides a specification of the research problem in this thesis. This specification is given in terms of the research goals, the research product and the corresponding research questions.

1.3.1 Research goal

The goal of this thesis is to contribute to the development of a theory (1.3.2) of requirements evolution by providing an understanding of the dynamics of requirements change in groupware implementation and use. To establish the usefulness of the research, the achievement of this goal is meant to contribute towards improvements in the practice of requirements engineering. In particular, on improving requirements elicitation processes and managing risks involved in deploying such class of applications in organizations, which are practical ends this research also seeks to address.

1.3.2 Research product

The envisioned research deliverable of this project is a theory of requirements evolution (Bacharach 1989; Whetten 1989; Elster 1998; Hedstrom & Swedberg 1998; Silverman 2000; Gregor 2006; Glaser & Strauss 1967; Yin, 2003) applicable to the domain of groupware system implementation. The reference to theory in this research is closer to the concept middle range theory in social science which is differentiated from covering laws in natural science (Hedstrom & Swedberg 1998). An example of middle range theory are social mechanisms which serve as intermediate between description and law (Elster 1998). They provide explanations but not predictions. Applied in this thesis, the theory developed is in the form of a conceptual framework that offers an insight into the basic character of the requirements evolution (Eddins 1967) through a system of concepts and statements. The concepts are meant to provide a description of the phenomena. The statements make use of the concepts to formulate perspectives on how to regard the phenomenon. It is not a theory in the form of causal relationships and covariation between variables and events. As a theoretical tool, the conceptual framework is aimed at learning about requirements evolution surrounding the implementation and use of collaborative technologies in organizations. Specifically, this entails seeking conditions and settings, i.e patterns, that depict requirements evolution in groupware use. The conceptual framework will also be a source of hypotheses which is meant to address the goal of contributing to the development of theory on IT and organizational evolution.

1.3.3 Research questions

The following research questions guide the conduct of this investigation.

Central question: How do requirements change in the evolutionary process of groupware implementation and use?

Supporting the central question are the following sub-questions (SQ):

SQ1. What are the impacts of implementing and using a groupware application on the requirements?
SQ2. What kinds of situations in groupware implementation promote requirements change?

SQ3. What mechanisms of change describe requirements evolution?

1.3.4 Definition of Terms

Before proceeding with the rest of this thesis, it is useful to first provide preliminary definitions of terms and concepts that are used and referred to in the succeeding parts of this book. These definitions are refined further in the course of this thesis when appropriate.

Groupware is considered to be any group-oriented ICT application that provides support for cooperative work. Cooperative work is hereby used to mean primarily unstructured work in which the team has some freedom in organizing the way how it is performed. It would include tasks such as communication, information sharing and document exchange, among others -- activities that normally performed when one is part of a group working on a common or shared goal. This way, groupware can be contrasted to workflow systems in which the order of how work is done is important and services are delivered following a structured, non-arbitrary and sometimes less flexible way.

Requirements. We will initially refer to requirements as desirable properties (Wieringa 2003). Since we are talking about a software product, these desirable properties are that of a software product. In a stricter definition, requirements are the demands ascribed to a software product. To cite an immediate example, providing real-time communication support for an internationally distributed team is a requirement for a groupware application.

Implementation is the process of introducing and making a groupware application for available for use in the organization. This is differentiated from the notion implementation in the context of software engineering where it is used to refer the coding and programming of the software product. In this study, implementation is regarded as a temporal concept where the completion of a software product or a working prototype that users can operate and work with is a necessary condition.

1.4 Research strategy and methodological plan

To approach the knowledge problem specified in the research goal and further elaborated through the research questions (specifically SQ4), we have chosen a research strategy in which first-hand access to primary sources of data is possible and direct observations of the phenomena can be carried out. Motivating this choice of research strategy is the earlier mentioned absence of empirical knowledge that aims to explain the nature and dynamics of requirements evolution which is a poorly understood phenomenon (See Section 1.3). In addition, the goal of contributing to theory development motivates this choice of strategy. Specifically, this strategy is carried out through a case study methodology (Eisenhardt 1989; Yin 2003; Silverman 2000). The specification of this methodology is given in Chapter 3. The cases investigated in this study serve as primary sources of data and information for deriving theoretical conclusions. These cases consist of actual deployment and implementation of groupware systems in organizations. See Chapters 4-7.

For the multiple comparative case study approach, a total of four cases of groupware implementations in organizations were investigated. The general mindset out of which these cases were investigated proceeded from an incremental outlook: the first case was investigated with an exploratory approach goal based on preliminary conceptual model. Updates from the first case were fed back to model and a second case was selected. With a more or less firm conceptual framework, another case study was conducted. Data gathering is done mainly through qualitative
means: interviews, document analysis, observations and artifact inspection. An overview of these cases and their descriptive characterization is shown in Figure 1-2.

![Diagram of case studies]

*Figure 1-2. List of case studies*

1.4.1 Methodological plan

On the basis of literature study, a preliminary conceptual framework was formulated with an open set of study propositions. For this, an initial, exploratory case study was conducted. Out of this initial case, a more refined conceptual framework of requirements evolution is formulated, thus setting the stage for a second study. The resulting framework is further validated through a third case study. A consolidated and comparative analysis of the cases is carried out confirm concepts, refine statements and finally to derive patterns of requirements evolution.

Drawing from Eisenhardt (1989) and Yin (2003), the implementation of this methodological plan is shown in the following flow (Figure 1-3).

![Diagram of methodological flow]

*Figure 1-3. The methodological flow adopted in this study*

1.5 Structure of the thesis

In the rest of this thesis, Chapter 2 introduces the preliminary conceptual framework by providing a discussion of the related work and literature. Chapter 3 lays out the methodological justification of this study and the operationalization of the conceptual framework. Chapter 4 presents the first case study and the first wave of updates to the conceptual framework. Chapters 5, 7 and 8 are the write ups of the remaining case studies. Chapter 6 is an interim chapter that presents the revised conceptual framework on the basis of the first two case studies, Cases A and B. Chapter 9 is a multiple case analysis chapter that aggregates the findings from all the cases. It provides the final update to the conceptual framework through the discovery of requirements evolution patterns, which is the most important contribution of this study. Chapter 10 gives the final remarks with a review of the research goals and questions, and reflections and directions for future work.
CHAPTER 2

THEORETICAL BASIS AND CONCEPTUAL FRAMEWORK

This chapter is a review of related work and literature. The goal is to develop a conceptual framework for describing and explaining requirements evolution based on a theoretical grounding of the research problem. For this purpose, we surveyed and reviewed literature on the following selected topics: requirements (Sections 2.1 and 2.2), requirements evolution (Section 2.3) and groupware (Sections 2.4 and 2.5). The conceptual framework including its considerations and next steps are discussed in Sections 2.6 through 2.8.

2.1 Requirements – what

The Oxford paperback dictionary defines the term requirement as something that is needed or compulsory. Usually, it is a prerequisite, something that is needed in advance in order to proceed, do, or build something. For example, a good knowledge of different didactic approaches is a requirement in teaching a course in the classroom. Or in a design effort (Dick et al. 2009), analysis is a prerequisite to design or to any form of interventions to improve a situation.

2.1.1 (Software) requirements: varying levels of interpretations

Therefore, when talking about software, requirements are the set of prerequisites specified in advance before software development or acquisition can proceed. In Chapter 1, we defined requirements as desirable properties of software (Wieringa 2003). Through these properties, the software is supposed to achieve its intended role in its operating environment. For software systems which are embedded in business and organizational environments, the desirable properties of these environments are also requirements for the system. These environments impose certain conditions upon the system. For example, budget, an existing infrastructure and platform, preference for a certain project management methodology, time and other resources. These impositions are requirements called constraints.

Given that a software project is a multidisciplinary effort involving different parties, requirements as desirable properties of the software and its operating environment may mean differently for different stakeholders involved. What requirements are depends on who you talk to. For the business manager, a requirement may be to increase the productivity of her unit. For the IT consultant, a requirement could be a new workflow chart for a new business process module in Enterprise Resource Planning (ERP) software such as SAP. Whereas for the ordinary user, her requirement is that she should be able to issue a receipt to the customer who lodged in the hotel upon check-out. From the perspective of these different stakeholders, a requirement manifests itself in certain ambiguous ways as a need and becomes articulated as a problem.
amenable to a software solution. It can be further refined such that it already provides a sketch to
a solution and written in paper to comprise what is called a requirements specification document.

The broad reference to requirements has been widely addressed in literature (Wieringa &
Heerkens 2006; Alexander 2002; Kovitz 1999; Gause & Weinberg 1989; Wiegers 2003;
Sommerville & Sawyer 1997; Lauesen 2002). Wieringa and Heerkens (2006) argue that the term
‘requirements’ is quite overloaded because it is used to refer to many things at the same time. For
example, Kovitz (1999) states that a requirement is a problem definition that contains information
that programmers and interface designers need in order to make the computer bring about effects
outside of the computer. For other authors, requirements are system services and the constraints
under in which it must operate (Maciaszek 2005; Sommerville & Sawyer 1997). It is also quite
common to add a qualifying description to the term requirements in order to get a better idea of
what is actually being referred to, i.e. user requirements, business requirements, system
requirements, data requirements, operating requirements, etc. In most occasions, the term
requirements, is used interchangeably with software specifications. For example, the commonly
cited DOD-STD-2167 updated later on to become MIL-STD-498 (US Department of Defense
1994), which specifies processes for developing and documenting mission-critical software
systems, is loosely referred to as requirements by contractors and researchers alike (Chung &

Lauesen (2002) proposes structuring requirements in a hierarchy in order to make
communicable to different stakeholders involved. There are 4 levels in the hierarchy:

- R1. Business goal level requirements,
- R2. Domain level requirements, which are the tasks to be supported,
- R3. Product level requirements, which are the specific functions of the software; and
- R4. Specific product specification in which product interface is given in detail.

2.1.2 More than just about software

The discussion (Section 2.1.1) points out that requirements are more than just about software.
Requirements encompass the environment in which it is used, especially in the case of
information systems. As mentioned, the business can impose conditions under which the software
must operate. In order to serve as a solutions to some real world problems, the software must be
grounded in the world outside the computer. This means taking into consideration the
requirements from the users and their context (Jarke & Pohl 1994; Jackson 2001; Grudin 1994;
Bannon & Hughes 1993).

2.1.3 Requirements as a continuum of problem definitions and solution specifications

Based on the previous discussions (Sections 2.1.1 and 2.1.2), it makes sense to regard
requirements as a continuum of problem definitions and solution specifications (Hull et al. 2005).
Requirements are problem descriptions and solutions specifications at different levels of focus,
scope, roles and sequence. Requirements are problem domain descriptions that provide inputs as
to what a software system should do in order to address the problem. For example, in a travel
booking context, it should be possible for travelers to personally book their flights without having
to visit or call a travel agent. Therefore, the travel booking system should be able to support this.
At the same time, requirements are also specifications of how a software system should realize
the needed functionality. For example, the application programming interface that would enable
analysts to load new data to the system should be available in Java because it is platform
independent. In most instances, the the distinction between the ‘what’ and the ‘how’ in terms of
requirements is rather blurred. In reality, it is hard to make this separation. Sommerville (1995),
citing an example by Davis (1993) demonstrates this point:
If a company wishes to let a contract for a large software development project, it must first define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering perhaps different ways of meeting the client organization’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.

Likewise, the broad reference to requirements in terms of problems and solutions can be regarded as a delineation between management and engineering concerns (Hull et al. 2005). High level system descriptions such as stakeholder goals and need statements are managerial concerns and therefore belong to the problem domain. Lower level requirements that capture global system requirements and solution specifications that provide design details belong to the solution domain as engineering concerns. Sections 2.1.4 and 2.1.5 elaborate on these two dimensions of requirements.

2.1.4 Requirements as problem definitions

As a problem description, a requirement is a formulation of a real-world problem. However, before a problem can be formulated, it follows that someone has detected or felt that a problem exists. For example, the manager of the Human Resources Department perceives an undesirable situation in her department. A problem is generally conceived as an undesirable situation that is significant and may be solvable by some agent, although with difficulty (Agre 1982). The key elements that characterize a problem are the gap between preferences and reality – what is perceived and what is desired – the importance of remedying this gap, and the expected difficulty of doing this (Gause & Weinberg 1989; Smith 1989). On the basis of this, a requirement is a specification of a goal that is to be achieved by an as yet unknown solution, i.e. a requirement characterizes the problem to be solved. For example, in a project to support traumatic brain injury patients (Hui et al. 2003), the need to maintain contact is an example of a problem formulation which is based on what patients want without immediately resorting to a software functionality specification. Eventually, such problem specification can be implemented through software.

Requirements also represent in very abstract terms as desires. Stakeholders, who are part of the problem, desire the gap to be reduced (Gause & Weinberg 1989). Verbalizing and externalizing the problem, especially to communicate it to others is to define it. Defining a problem can be done in various ways. Smith (1989) had identified various conceptualizations of a problem definition. On the basis of these, a problem can be defined or formulated by giving statements that reflect any of the following:

- Gap specification: comparing existing and desired states, i.e. what ought to be vs. what is.
- Difficulties and constraints: identifying factors inhibiting goal achievement
- Ultimate values and preferences: stating the final ends served by a solution
- Goal state specification: identifying the particular goal to be achieved, i.e. unresolved customer complaints should be reduced by 10%.
- Means and strategies: specifying how a solution might be achieved
- Causal diagnosis: identifying the cause(s) of the problematic state
- Knowledge specification: stating facts and beliefs pertinent to the problem
- Perspective: adopting an appropriate point-of-view on the situation.

All these descriptions can be arrived at through the conduct of a systematic process called problem analysis. Problem analysis includes domain analysis, which is an attempt at understanding the nature, characteristics of problem elements and phenomena unique to a
particular domain, i.e. accounting for accounting systems, social communication for communication and coordination systems (Jackson 2001).

One desirable property of a problem definition is that it should promote effective problem solving (Smith 1989; Dery 1984; Schön & Rein 1977). One way of doing this is to further structure the problem, i.e. decompose it into smaller sub-problems. This can be also by framing and re-framing the problem (Schön & Rein 1977; Jackson 2001; Wieringa 2003). One popular technique of structuring problems in the software development problems is goal decomposition (van Lamsweerde 2001; Van Gundy 1988; Anton & Potts 1998). Problems are defined as specific goals to be achieved and these are further decomposed into sub-goals. Through these decompositions, a problem is made amenable to solving.

To conclude, as problem descriptions, requirements can be expressed as problem definitions, i.e. verbalization and externalization of an identified problem, or as problem decompositions, i.e. a structured version of the problem definition, in which the problem is made ready for solution. In the latter process the complex interrelation between problems and solutions arise: what comprises as a solution for someone, i.e. the business manager opting for an ERP implementation is a problem for the IT Manager who will then choose, evaluate and specify what kind of ERP application is needed by the organization.

2.1.5 Requirements as solution specifications

As a solution description, a requirement is a specification of one possible solution to a given problem. Usually, the solution is hereby understood as software. As a description of a solution implicating software, a requirement of this kind is commonly known as a software requirement specification. A software requirement specification contains prescriptive information about the software to be developed or to be acquired, i.e. what should the software do, what it should have, what should be its properties, etc. These specifications for software are usually given in the form of: functional specifications, quality specifications, data specifications and constraints.

Functional specifications

Functional specifications are the things a software system should do and accomplish. These are the functions of software – what it should bring about in the environment by virtue of its programming (Kovitz 1999). As specifications, they prescribe the behaviour of software. Other authors prefer to call functional specifications system services or tasks that the software would do (Wieringa 2003; Lauesen 2002). This is because the software as solution is supposed to deliver some sort of support to its environment. For example, a software system should assist a hotel receptionist to book guests and assign rooms to the guest (Lauesen 2002). Because software systems have to deliver some sort of support to its operating environment, its functions contain some information about the domain. This way, the functional specifications correspond to existing tasks or business processes in an organization. These result into what we can term as functional modules in larger software systems, where these modules are further broken down into smaller functions and sub-functions. Consider the following example of a module called Treasury in the Financial package of SAP R/3 ERP software adapted from a study conducted by Rolland and Prakash (2001) (See Figure 2-1). The Treasury module is meant for medium and long-term financial planning along with management and control of revenues. The authors have identified two built in goals in the modules: Plan Funds and Manage Liquidity. Under the Plan Funds goal, other sub-functions form its refinement such as Forecast Budget Item, Construct Budget. In this example, the ERP application’s functional module corresponds to the user department’s main responsibility and the sub-modules around it correspond to the smaller tasks the department has to perform.
Functional specifications can be written or composed as plain text, structured text or in terms of a diagram with formal notations, i.e. UML. Accordingly, the specifications can be given at various levels of abstraction, i.e. one function can be an aggregation of smaller functions corresponding to one complete task or process, or a function can be an atomic specification of one task.

**Quality specifications**

Also called non-functional requirements, quality specifications are descriptions of how well a system must perform its functions. There is actually a preference towards the term quality requirements instead of non-functional requirements because these requirements for software do not mean that they are not functioning. Instead these requirements specify the desirable attributes of the functions that make up the software.

When identifying quality specifications, several authors refer to several classifications of factors indicating software quality (Lauesen 2002; Pressman & Ince 2000). A list of commonly referred quality properties of software are given as follows (McCall et al. 1977; Grady & Caswell 1987; ISO 2001; IEEE 1998):

- Maintainability: the ease with which errors are found and repaired
- Correctness: the extent with which the software satisfies the specifications
- Performance: also called efficiency, it is the degree to which the software makes optimal use of resources
- Reliability: it refers to how well the system can be depended upon to perform its functions properly, i.e. how often is it available and how frequently does it malfunction
- Usability: the effort required to learn the software especially for its intended users
- Security: also called integrity, it refers to the extent to which access to data by unauthorised users can be controlled
- Portability: effort required to transfer program from one hardware and/or software system environment to the other
- Interoperability: the compatibility of the software system with the existing systems in which it must interface with

It is also important to note that quality requirements do not always pertain to software only. It can also refer to the desirable attributes of the supplier or the developer of the software. For example, in software maintenance efforts, it can be stated as a quality requirement that a software engineer be made available within 24 hours when the system encounters an unnecessary downtime. Or that the supplier is supposed to provide 120 hours of training to the operators of the software.

For Commercial-Off-The-Shelf (COTS) software, it has been observed that competing products have, more or less, the same kind of functionalities. Therefore, these are hard to
differentiate on the basis these. However, it is also been observed that what differentiates one COTS product to a competing one are the quality factors that is embedded in the packaging of the software, i.e. maintainability factors such as the support that is given after the purchase of the software (Maiden & Ncube 1998). In other words, this is the aspect where one COTS vendor-developer gains a competitive advantage of the market.

Data Requirements

When specifying a solution in the form of software, part of what is to be specified is its data requirements. In order to produce an output, a software system must process some sort of inputs. These are usually in the form of data that the system must process or store. Data requirements are the necessary pieces or fragments of information that are needed for the system to perform accurately or to bring about its desired effects. These data pieces usually have a structure which can be modeled through a class diagram.

Constraints

Any problem solving endeavor through software development is bound by a set of constraints. Limitations in resources, e.g. budget or time, put restrictions on how the solution is to be arrived at. Constraints limit one’s freedom in designing the solution. Constraints can be given in the form of a chosen language development standard, policies for database integrity, resource limits, operating environments, etc (IEEE 830). For example, a requirement that indicates that the software has to be developed using Lotus Notes is a constraint.

2.1.6 Expression and documentation: tacit requirements

Requirements come from stakeholders and their context (Lauesen 2002; Gause & Weinberg 1989; Beyer & Holtzblatt 1998). The ideal situation is that these requirements are made explicit and communicated in a software requirements specification document i.e. DOD-STD-2167A, IEEE 830 (IEEE, 1998). For large software projects, a software requirements specification document serves as the contract between two parties – the buyer of the software and its vendor or developer. On the other hand, smaller companies and projects have their own templates for writing requirements or documenting their projects.

Written specifications could be given in the form of natural language, structured text or a diagram with formalised notations and semantics, i.e. UML. Usually, it comes in the form of a document. On the other hand, as requirements are also desires people have, it is often not possible to externalize and make them explicit. If not verbalized, this desire means it is a tacit requirement (Gause & Weinberg 1989; Lauesen 2002). Tacit requirements are requirements too because they have an impact later on. It is generally known that users find it difficult articulate their needs or task problems. It is also hard for them to visualize what they want in a software solution if they haven't seen it yet.

2.1.7 Summary

To summarize, the requirements can be problem descriptions or solution properties (Sections 2.1.3 through 2.1.5) and they can describe software properties or business properties (Section 2.1.2). They can be documented and tacit (Section 2.1.6).

2.2 Requirements process - how

Establishing what the requirements are involves a process, which is also but one part of the larger software engineering cycle. Requirements are crucial inputs in the design process. In fact, a good indicator of a software product’s overall quality is how well it fulfills the requirements.
Therefore, in order to come up with a software product for the business means that requirements have to gathered, analyzed, communicated and agreed upon. Having such a process or structure in a project can contribute to its success (See Table 1-1 & 1-2). For example, Blackburn et al. (2000) had found out that more time and effort spent in the early stages of software development yield faster cycle times and higher productivity. Likewise, more intense attention given to users and customers has also been observed to lead to more successful software development projects such as decreased number of iterations (Keil & Carmel 1995; Chatzoglou & Macaulay 1998).

There is a general agreement that lack of attention to or a superficial treatment of this process has been attributed to as the major reason as to why IT systems fail in meeting expectations and goals regardless of the application domain (Lauesen 2002; Gause & Weinberg 1989). These failures are reflected in terms of missing functionality, ill-defined system mission, poor interface and inefficient user and task support.

2.2.1 Requirements engineering

The gathering, analysis and utilization of requirements for software are the research and practice specializations of the Requirements Engineering (RE) discipline. The annual international IEEE Conference on Requirements Engineering (http://www.requirements-engineering.org) states that RE is a branch of systems engineering that is concerned with the goals, desired properties and constraints of complex systems, ranging from embedded software systems and software-based products to large enterprise and socio-technical systems that involve software systems, organizations and people.

Requirements engineering forms an integral part of the software development process. Every software lifecycle model incorporates requirements engineering or a counterpart process thereof, as a front-end activity with several cycles of iteration in the design and development of software systems (Pressman & Ince 2000; Pfleeger 1999, see ch. 2; Kruchten 2004; German Ministry of Defense 1992). Specifically, these activities are aimed at discovering, learning, understanding and maintaining the goals, functions, and quality properties of a system (Zave 1997; Kotonya & Sommerville 1998). In the classical waterfall model of software development (Royce 1970), requirements engineering, referred to in the model as analysis, precedes design and development. The outputs of this analysis, namely, requirements serve as inputs to the design process.

It has been pointed out that the engineering in requirements engineering need not to be construed as a process of building and constructing as it is understood in terms of the conventional engineering disciplines (Sommerville 1995). Rather it implies the use of systematic and repeatable techniques to ensure that system requirements are well understood and properly specified (Kotonya & Sommerville 1998). It is for this reason that other terms such as requirements process, requirements analysis or requirements analysis process are used by others (Robertson & Roberston 1999; Beyer & Holtzblatt 1998).

2.2.2 Requirements engineering activities

As a process, requirements engineering can be broken down into a series of activities and rational steps that also incorporate an iterative structure. These activities include (Nuseibeh & Easterbrook 2000; Kotonya & Sommerville 1998; Lauesen 2002):

- Requirements elicitation

Requirements come from customers, users and other stakeholders of the system. Requirements elicitation is the initial step in the requirements that is aimed at learning, discovering and surfacing the needs of stakeholders (Lauesen 2002). The role of the different stakeholders in this activity is central. In this activity, the requirements engineers together with the stakeholders discover, review, articulate and understand the user’s needs as well as the constraints on the
software and development activity (Thayer & Thayer 1994).

There are several techniques suggested in literature on how to carry out a requirements elicitation activity. Accordingly, there is an increased emphasis on the use of techniques derived from the social sciences, such as ethnography (Hughes et al. 1994) and contextual inquiry (Beyer & Holtzblatt 1998). Likewise, there are also approaches that give weight to task analysis (Lauesen 2002). Such reference to socially oriented techniques in doing requirements elicitation is largely motivated by the need for an in-depth understanding of the social world domain in which the system will be implemented. A good understanding of the operating environment translates to a good understanding of what the requirements are.

• Modeling and analyzing requirements

Usually, observations derived from requirements elicitation are informal and raw. These need to be modeled and analyzed in order to check for requirements conflict, inconsistency and incompleteness.

• Communicating requirements

Communication with stakeholders comprises a very important part in the requirements process. Requirements are communicated back to the stakeholders in order to secure a common understanding. Requirements are likewise verified through regular communication with stakeholders.

• Agreeing requirements

Agreeing requirements often involves the process of negotiation to resolve conflict, priority setting and conflict management. Sometimes stakeholders have very high expectations about a system and it can happen that these are not realizable due to certain constraints. It is important that these are communicated properly and resolved.

• Specification

This activity is concerned with the formalization of the requirements into written form. Formalization implies the transformation of the informal observations, models and processes into precise and unambiguous specifications. Documenting the requirements in this manner also facilitates traceability and management.

• Evaluating requirements

The RE process is not terminated once the specifications are written and implemented in the design of the system. Validation is an activity in which the system is evaluated against the demands of the users and stakeholders to ensure that each part of the program is wanted (Lauesen 2002). Along the way, some requirements creep and evolve and one way of evaluating requirements is to assess how the system is actually being used.

• Managing and maintaining requirements

In the same manner that software has to be maintained so do requirements. Requirements have to be managed and maintained for purposes of addressing issues such as consistency and requirements creep, which was mentioned earlier and for anticipating evolution and software adaptation. For these purposes, software support is available such as DOORS or Rational (http://www-01.ibm.com/software/rational/)

In practice, these RE activities are closely interwoven and the flow of activity from one step to the other is very fluid. RE is a highly iterative and evolutionary process in which the activities are not bounded to a sequential one-shot effort (Pfleeger 1999). What can also be hinted from the description of activities above is that a great deal of the RE effort is non-technical. RE is very social activity in which communication and understanding of social processes are very important.
There is no one best way of doing RE nor is there one ideal RE process that fits all instances of problem solving. What remains invariant however is the idea that in order to build or acquire systems that the intended people will use, and like to use, it has to be founded on a proper understanding of needs, tasks, goals and operating contexts.

2.2.3 RE approaches and techniques

There are various approaches and techniques towards doing requirements engineering. Some of these incorporate a complete process of requirements engineering, i.e. from elicitation to specification and management, while some focus on a certain requirements engineering activity, i.e. elicitation techniques while some focus on notation. A broad classification of these is given as follows:

Complete RE process approaches with incorporated methods and modeling notations

These are approaches that have a fully developed methodology, i.e. rules, procedures, heuristics and steps to be followed for conducting requirements engineering. In some instances, these approaches have their own set of techniques and notations for modelling, i.e. UML. Because they are holistic in their approach, these serve as a resource for those who would like to conduct a requirements process – analysts, requirements engineers, consultants. Examples of these complete approaches include the Volere Method (Robertson & Robertson 1999), the Rational Unified Process (Kruchten 2004), Contextual Design (Beyer & Holtzblatt 1998), among others.

Approaches that form part of a larger software development method or vice versa

As requirements engineering forms part of a larger software engineering process, some RE approaches are subsumed by a larger software development methodology. Therefore they are not loose or de-coupled from a software engineering process. Stated another way, integrated software development approaches They fulfil the requirements gathering and analysis part of software development. Examples of these approaches include: Joint Application Development (JAD) and Rapid Application Development (Pressman & Ince 2000).

Techniques that focus on specific requirements engineering activities

In most instances, RE approaches focus on one particular requirements activity as its core specialty, i.e. elicitation, modelling, negotiation, etc. This way, they serve as techniques for conducting a certain requirements activity. A number of these techniques are focused on requirements elicitation because it is considered to be one of the critical process of RE. This is the part where requirements are gathered from stakeholders and therefore it is important to have the right requirements and these should be complete. Other techniques focus on the communication aspects of the process, i.e. negotiation and prioritisation of requirements. Likewise, there isn’t a dearth of techniques that focus on the modelling part of requirements engineering. It is also important to keep in mind that some of these techniques also form part of a larger RE methodology, and eventually software engineering methodology.

Examples of requirements elicitation techniques which encompass other activities that involve the social process domain, i.e. communication, negotiation, and prioritisation include: contextual inquiry, which is a part of the larger contextual design methodology (Beyer & Holtzblatt 1998), ethnography-based approaches (Hughes et al. 1994), participatory design (Kuhn & Winograd 1996), ETHICS (Mumford 1995), Soft Systems Methodology (SSM) (Checkland & Scholes 1990) as well as other generally known data gatherings techniques such as interviews, questionnaires, focused group and brainstorming sessions. A survey of other techniques that focus social issues is given by Goguen and Linde (1993), as well as how experts are known to do it is
provided by Hickey & Davis (2003).

**Approaches that proceed from a certain school of thought or conceptual frameworks**

There is a general distinction found in literature about two schools of thought on how requirements engineering should be done. One of this is the software engineering and project management arguments that focus of thought that focus largely on structured methodologies. We have already cited earlier a few example of these approaches, i.e. Rational Unified Process. The other school of thought can be lumped under the socio-technical perspective which incorporates social and humanistic orientations towards design. Within this perspective, techniques are borrowed from the social sciences are used, such as ethnography. Some of the examples we have also given earlier incorporate this perspective, i.e. contextual inquiry, ETHICS, etc.

**Software support for RE processes**

Various software applications as support for RE incorporating a certain methodological approach are also available. For example, the Rational Unified Process provides fully automated support for the software engineering process which subsumes the requirements engineering processes.

2.3 Requirements evolution

Requirements evolution is the change in requirements over time. The change can consist of qualitative and quantitative aspects of requirements. For example, the specifications have increased in terms of number or a specification of one requirement has become more precise. It can also be that a tacit requirement has become more explicit or that one specification is no longer wanted and therefore it is discarded.

Requirements evolution is a persistent phenomenon in software design and use. This is implied in most software engineering lifecycle models. All throughout the phases of software system’s life, requirements continue to evolve. Stated in another way, requirements span the two modes of interactions human beings have with a technological artifact, namely design and use (Orlikowski 1992). Where these two modes can be situated in a time continuum, requirements persist in this continuum. (Figure 2-2).

![Figure 2-2. Modes of interaction with a software artifact and the persistence of requirements](image)

Requirements evolution is a problem in software development. One generally known problem associated with requirements evolution is requirements creep, also called feature creep, which is the tendency for requirements to multiply during the later stages of the process. This is due to clients changing their minds and of developers aiming at improving the software. All these changes have impact on the cost as well the delivery of the product (Boehm, 1981; Ewusi-Mensah 2003). At the same time, the impacts of requirements evolution to a software project’s success are known to be significant. Sections 1.2.1 and 1.2.3 identify changing requirements as one of the factors that lead to failed projects. The specific examples of failed projects that we
have enumerated all have requirements issues at one point or the other that led to their failure.

Requirements evolve mainly as a result of increased understanding of the problem by the designers and of the users themselves (Lutz & Mikulski 2003, 2004; Sommerville 1995). During the later phase of the cycle when the system is used, this increased understanding is given in the form an expanded technological frame where users have acquired a more profound appreciation of the functionality of software (Orlikowski 1996). These are examples of E-type software systems that evolve because their use changes the world in which they are embedded enough to change their own requirements (Lehman et al 2000, 1985). Other factors also prompt requirements to evolve for example new legislation, changing business environment and latest technological breakthroughs.

On the other hand, the reference to requirements evolution as a problem in software development is mainly associated in the design part where a software product is undergoing development. Nevertheless, requirements change too, after a system has been developed, deployed and used organisations. Servicing these requirements changes after software delivery is software maintenance. There is considerable evidence which indicate that the bulk of software maintenance activities is spent on major design work, i.e. 80% is spent on adaptive, perfective and preventive maintenance activities, and less on corrective maintenance, i.e. fixing bugs, which accounts for only 20% of software maintenance efforts (Munro 1989; Frazer 1992; Rajlich & Bennett 2000; Pressman & Ince 2000). For this reason, design activities are proposed to be iterative and continuous. Specifically, Hughes and colleagues (1996) argue that software design issues do not cease with the initial specification of requirements, but persists when systems have been introduced into their settings.

2.3.1 Requirements evolution: a poorly understood phenomenon

Requirements evolution during the latter phases of a system’s lifecycle, namely deployment and use, is a poorly understood phenomenon (Section 1.2.4). This means that there is insufficient knowledge about requirements evolution, i.e. empirical descriptions of its dynamics, how it takes place and what are the factors that cause it. The current practice in requirements engineering tend to focus on how-to approaches that are aimed at managing and supporting the evolution process. To date, the landscape of research into requirements evolution can be classified into two dominant research themes. These are researches covering the following efforts:

- Approaches and methodologies dealing with how to manage, control and mitigate the impact of requirements change and evolution (Barber et al. 2002; Carter et al. 2001).

These are the leading type of research efforts on requirements evolution. These efforts provide solutions on how to address requirements evolution as problem. One example of such is EPRAM, which stands for Evolutionary Prototyping with Risk Analysis and Mitigation (Carter et al. 2001). It extends the evolutionary prototyping paradigm in software engineering by incorporating risk identification and assessment at each prototyping activity and at the end of each prototyping cycles.

- Descriptions and classifications of software specifications evolution
  (Harker et al. 1993; Lubars et al. 1993; Sommerville 1995; Dobson et al. 1993)

These research efforts on requirements evolution proceed from the notion of requirements as solution specification. Usually, these are analyses of requirements – software specifications, which have been identified and documented. That’s why their evolution can be traced and described. Descriptions are given in terms of categories and taxonomies of requirements and the types of changes. For example, Sommerville (1995) broadly classifies requirements into enduring and volatile requirements, which is a very simplistic classification and does not provide
additional information evolutionary properties of requirements.

Another classification is given by Anderson & Felici (2002) categorized eight types of requirements change based on changes made to requirements such as addition, deletion or modification, using data from the different software releases made for an avionics safety-critical system. These investigations also shed light on the properties of change such as which requirement change type is most prevalent over time or what is the average rate of change. In an earlier study, Harker et al. (1993) likewise presented a similar classification with an attempt to provide a deeper understanding of requirements evolution by relating the requirements types to users and organizations. This classification, which Sommerville (1995) collapsed into four types, can be further summarised into three types (Table 2-1):

<table>
<thead>
<tr>
<th>Requirements Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable requirements</td>
<td>Requirements that change due to the changes in the environment in which the system operates, i.e. new law, new business rules</td>
</tr>
<tr>
<td>Emergent requirements</td>
<td>Requirements that emerge due to an increased understanding of the system by the customers. This could be during system development or when users have begun interacting and using the system.</td>
</tr>
<tr>
<td>Interoperability requirements</td>
<td>These are requirements that are dependent on existing systems or standards, which continually change.</td>
</tr>
</tbody>
</table>

*Adapted from Sommerville (1995) and Harker et al. (1993)*

The main difference between emergent requirements and mutable requirements in this classification is that the former is independent of the software system. Changes in the operating environment take place autonomously regardless whether a software system is in place or not. On the other hand, emergent requirements are those brought about or are under the influence of the software system in place. Interoperability requirements can also be regarded as compatibility requirements and these are most usually classified as quality requirements.

While with these approaches it is possible to know that requirements have changed and what kind change on the requirements took place, i.e. addition, deletion, modification has taken place, these however have little to say about why requirements have changed and how these evolved. An added limitation of these approaches is that they focus on requirements that were explicit, while requirements that were unarticulated, tacit requirements, evolve too.

On the other hand, studying requirements evolution in the context of software deployment and use remains to be a challenging task. The complex interaction between the social environment and the software system adds extra complication, which makes the identification and specification of requirements changes elusive and difficult. As a system is introduced for use in an organization, the demands for this system also change. Organizational change resulting from ICT implementation is one instance where requirements change due to the introduction of the system itself.

### 2.3.2 Requirements evolution as a dynamic process

Recent studies, which do not fall in the typical mould of research efforts about requirements evolution, try to provide empirical knowledge about the dynamics aspects of requirements change. An important contribution in this area is the work done by Lutz & Mikulski (2003) which shed light on these through the identification of common requirements discovery and resolution mechanisms from the testing up to the deployment phase i.e. operational use, of a mission-critical system. Their study also confirmed that requirements continue to be discovered during the operational use of software and measures are taken to resolve the changes. They based their
analysis by studying anomaly reports during these later phases of the software lifecycle. In order to get an idea of the dynamic aspects of evolution are, we provide the common patterns of change that they have observed as follows:

- Mechanism 1: Incomplete requirements resolved by changes to software
- Mechanism 2: Unexpected requirements interactions resolved by changes to the operational procedures
- Mechanism 3: Requirements confusion on the part of the testing personnel, resolved by changes to the documentation, and
- Mechanism 4: Requirements confusion on the part of the testing personnel, resolved by a decision that no change was needed

2.3.3 Continuing view of requirements engineering

Because of the persistence of requirements change over time all through a system’s lifecycle, it is proposed that requirements engineering ought to be a continuing process (Jarke & Pohl 1994; Mylopoulos 2009). This means that requirements engineering activities have to be carried out after a system has been developed and deployed in a user environment. This is consistent with the continuing view of design where design activities do not cease upon the completion and delivery of a software system to the client or users. This view is in contrast to the traditional view of requirements engineering as a front-end, sequence-bound activity in software development, and where work on the software at post-deployment phase is considered maintenance.

2.4 Groupware: an overview

Section 1.3.4 provides a preliminary definition of groupware as any ICT application that provide support for cooperative work in which the manner of working is not specified in advance. Detailed discussion about the nature of cooperative work is discussed in Section 2.5. Collaboration technologies, cooperative systems, coordination tools, group support systems, etc. are synonymous with groupware technology. However, for purposes of consistency, we have chosen to use the term groupware to refer to this class of applications (Andriessen 2003; Orlikowski 1996; Hinsen 1998).

Groupware systems are used to support and carry out the social domains of work, i.e. the aspect of one’s job in which interpersonal interaction and cooperative processes take place. These are activities that are basic to most tasks but are not officially specified as part of one’s tasks. They are not documented or prescribed in operational manuals. For example, someone sends an email to a colleague asking a favor: if she can get a copy of the presentation he gave the other day because it contains a nice presentation template that she can use for her report.

Groupware systems are also used by groups of people for a shared goal. Groupware technology is cooperative technology. People could use shared workspaces for the goal of sharing knowledge, they use email with the goal (one hopes) of getting work done, and they use videoconferencing with the shared goal of conducting a meeting. Actions like these which indicate cooperation are informal, intermittent, synergetic and largely unstructured. These properties make them difficult to specify in advance. Consequently, these are the activities groupware systems are meant to support.

Our study focuses on this application domain as the area of study. The main goal of this section and the succeeding section is to conceptualize groupware as a software application from which requirements evolution can be observed and reasoned about. As ICT applications continue to multiply over the years of which a number of these can be classified as groupware, it is first necessary to establish a clear understanding of groupware is.
2.4.1 Towards an evolving understanding of groupware

Over the years, the notion of what a groupware system is has evolved and continuous to evolve (Andriessen 2003; Grudin 1994, 2001). Originating as an extension of personal computing due to advances in network technology, groupware applications have evolved from simple interpersonal communication software to complex collaboration systems. In fact, the reference to groupware as a generic term for a wide-range of applications that support cooperative work is an appropriate development. It reflects the non-exclusivity of software functionality that support group processes which other software applications, not necessarily groupware, i.e. Enterprise Resource Planning, Customer Relationship Management, also contain.

The growth and evolution of groupware technologies can be further appreciated when situated in the larger context of different human structures and software development efforts that support each structure (Grudin 1994, 2001) (see Figure 2-3).

Figure 2-3: The context of groupware (Grudin 1994; 2001)

From this classification, there are four levels of societal structures which software systems can be built for: individuals, groups, organizations and communities. Individuals are single users of systems and are supported by way of stand-alone and personal productivity computer applications. Groups are collections more than one individual that have interaction with one another. Technological support for this kind of users is provided through network applications, workstations and groupware. Organizations are formal social entities made up of people and groups share a certain goal (Daft 1998). Organizations were largely and originally the object of most systems development projects and applications consisting of mainframes, corporate information systems, intranets among others (Grudin 2001). Communities are voluntary aggregation of people and groups, usually geographically distributed, who share a common interest but with no rigid structure imposed. The Internet and its suite of applications such as the WWW are facilitative technologies that provide support for communities.

It can be surmised from this classification that the context of groupware systems begins when the social milieu of use shifts from the individual towards groups and other larger societal structures such as organizations and communities. In the latter type of structure, people and groups come together. On the other hand, the shift from a personal stand-alone productivity tool to a complex information system for an organized whole, i.e. group, organization, community, the instantiation of a groupware application becomes diverse. Group-orientation according to Greenberg (1991) is key factor from which to distinguish groupware systems from other kind of systems. This way, isolated personal computers and mainframe systems or data warehouse in this sense are excluded from the concept of a groupware. As organizations and communities are also composed of groups and that systems built for these human structures also require groups of people to work together, overlaps exist. In-between are many-kind of group-oriented systems in which there is no rigid dividing line (Bock 1991; Ellis et al. 1991).
Considerable attention has been given to characterizing groupware applications (Johansen, 1988; Put 1996; Bock 1991; Greif 1988; Orlikowski & Hofman, 1997; Andriessen 2003; Hinssen 1998). This is because the general reference to ICT applications that provide support for cooperative work makes the task of recognizing whether an application is groupware, very challenging. As discussed in the previous paragraph, there is a blurred distinction between applications that support multiples of users in organisations and communities. All these applications are potentially groupware. This way, the application class of groupware could point to a large number, if not, infinite set of software products. It can be the case that either that all applications as long as it is not personal, stand-alone software, and are used by multiple people for related purposes could be groupware. For this reason, several authors have put forward several qualifying properties meant to distinguish groupware applications. For example Andriessen (2003) and Orlikowski & Hofman (1997) share the view that a groupware application is one which provides support for human communication, coordination and collaboration through software services such as information, shared repositories, discussion forums and messaging. On the other hand, Ellis et al. (1991) provides a more restrictive characterization in the sense that groupware applications are only those that have facilities such as shared workspace and that users have a common goal or task.

The general conclusion we can draw from these discussions is that there is actually no official standard definition of groupware that everyone must adhere too. This is by no means a disadvantage. In fact, it makes groupware a flexible application domain and a rich, promising and interesting area of research. In the computer supported cooperative work (CSCW) research community, examples of groupware tools used to report empirical findings on point to a wide range of software applications: from virtual environments (Drozd et al. 2001), meeting and videoconferencing systems to air traffic control (Hughes et al. 1992; Juhlin & Weilenmann 2001) and hospital information systems (Reddy et al. 2001) to real-time messaging systems and mobile applications (Handel & Herbsleb 2002). These approaches towards studying and instantiating groupware underscore the flexibility surrounding groupware applications and that the blurred distinction between applications does not impose a restriction. In other words, groupware is not a proprietary name for a system containing only groupware functionalities.

2.4.2 Groupware applications as socio-technical systems

Underlying these various characteriations is the socio-technical view ascribed to a groupware system. While a groupware application is in the first instance a technical system, it also has a social dimension. This is because the environment in which the software operates is a social system composed of people, processes and tasks (Mumford 1995). Therefore it is more apt to refer to it a socio-technical system. Correspondingly the social environment makes demands on the software system by making use of it and seeking possibilities and transforming it (Carroll & et al. 1991).

Viewed as technical system, a groupware is a technological product in the form of an electronic or digital tool with facilities such as email, shared workspaces, shared access to databases and repositories, messaging and discussion boards. Embedded in a social system, a groupware system, a groupware application enables interpersonal cooperative processes across time and space. It provides technical functionality for communication, information sharing and coordination. (Grudin 2001; McGrath & Hollingshead 1994). Specifically, in terms of Andriessen (2003), groupware systems can be distinguished from other ICT applications by having functions that serve the following human interaction processes:

- communication, i.e., exchanging signals
- cooperation, i.e., working together, making decisions
- coordination, i.e., adjusting the work of group members, leadership
• information sharing and learning, i.e., exchanging, sharing information and knowledge
• social interaction, i.e., group maintenance activities, developing trust, cohesion, conflict handling, reflection.

In addition, the behavioral processes surrounding the use of a system contribute to the 'groupware-ness' of that system. A general-purpose cooperative system such as email is a groupware in the technical sense; however it may be used for mass marketing purposes in which support for collaborative activities is absent. It is for this reason that the authors we have cited above (Ellis et al. 1991) draw on the use aspect of a system as a determining factor in contributing to the 'groupware-ness' of a system. According to them, for a system to be considered as groupware, a common task or goal must be present.

Greif (1988) mentioned that in the long run, the differentiation of a segment of the software product market in terms of groupware might not make much sense at all; all software will have the required features to support group use when appropriate. In this manner, our research studies the evolution of requirements for groupware functionality, which could be the functionality of any software product.

2.4.3 Groupware vs. workflow applications

Groupware applications can be differentiated from other networked and group-based applications such as workflow (Table 2-2). Although in Andriessen's (2003) framework, workflow systems are also collaborative technologies, we would like to consider workflow systems as a separate class of software applications. This is because the application domain of workflow is the opposite to that of groupware. Where workflow systems are tied to business processes and tasks whose executions need to follow a structured sequence, groupware applications on the other hand support cooperative tasks whose execution are not prescribed. In most occasions these tasks are performed in an informal manner and not immediately explicit.

Table 2-2. Distinction between groupware and workflow

<table>
<thead>
<tr>
<th>Groupware</th>
<th>Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperative work supported is not prescribed or specified in advance</td>
<td>Work processes supported follow a structured sequence; activities are</td>
</tr>
<tr>
<td>Work processes are informal and not readily explicit; eludes formal</td>
<td>routine tasks</td>
</tr>
<tr>
<td>analysis</td>
<td>Work processes are formal and explicit</td>
</tr>
<tr>
<td>Can support primary processes but users have freedom to voluntary</td>
<td>Users make use of the system because using the system forms part of</td>
</tr>
<tr>
<td>use the system unless strictly mandated; Breakdown in the software does</td>
<td>their work, and is tied to the regular operations of the organisation;</td>
</tr>
<tr>
<td>not lead to work or business stoppage</td>
<td>Users are ‘forced’ to use the system</td>
</tr>
<tr>
<td>Users can work their way around if the system is not available</td>
<td>Breakdown in software can have severe repercussions on the work; can</td>
</tr>
<tr>
<td></td>
<td>lead to stoppage of work or business</td>
</tr>
</tbody>
</table>

Another key distinguishing feature is that groupware supports tasks that are essential but non-critical in nature. By non-critical, we mean processes that are not strictly bound to primary business processes in a way that a breakdown in the process will not lead to a stoppage in the business or work. When a workflow system representing the primary processes of the business breaks down, the system or the business stops functioning. For example, in Case Study C (Chapter 6), it can be seen that when the course management system became unavailable, the teachers resorted to the manual way of communicating and disseminating information to students.

While groupware applications can also support primary processes, there is some amount of flexibility left in performing the task or that there is a way to work around it. In a way, for groupware systems, most users have the freedom to make use of the system, either voluntarily (Case A and C) or as a product of organisational mandate (Case B and D). Making this distinction between groupware and workflow in terms functional purposes has implications not only about how software requirements are gathered but also with how these requirements evolve. Therefore these two classes of software applications merit separate domains of investigations.
2.4.4 Redefining groupware

In this thesis, we define groupware as any ICT applications that provide support for cooperative processes of groups of individual working together, where these processes have no structure, or rather, they processes evolve a structure that is not specified in advance but arises spontaneously.

2.4.5 Examples of groupware application

Based on the above definition, there are a number of software applications and products classifiable as groupware on the basis of their functional properties. To give examples of groupware applications and products, it is useful to refer to a classification taxonomy of groupware. This is because there are also different kinds of groupware systems with different functional properties. Existing taxonomies are derived from a time-space dimension classification, or from application domain or from work content, among others (Ellis et al. 1991; Put 1996; Coleman 1997; Johansen 1988). Table 2-3 combines all these taxonomies into one and provides listing of application domains and tools that can be classified as groupware.

Table 2-3. Examples of groupware applications and software products

<table>
<thead>
<tr>
<th>Application Class</th>
<th>Electronic communication and messaging systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>Applications for exchanging messages, regardless of time differences.</td>
</tr>
<tr>
<td>Sample applications and products</td>
<td>Email Bulletin Boards Discussion Boards Microsoft Outlook; Lotus Mail; Pegasus Mail O'Reilly Web-Board; Lotus Notes</td>
</tr>
<tr>
<td>Application Class</td>
<td>Group calendaring, scheduling and coordination systems</td>
</tr>
<tr>
<td>Definition</td>
<td>Applications for calendaring, agenda planning, meeting and resource allocation.</td>
</tr>
<tr>
<td>Sample applications and products</td>
<td>Microsoft Outlook; Lotus Notes Organizer; The Coordinator (Winograd 1986)</td>
</tr>
<tr>
<td>Application Class</td>
<td>Decision support and electronic conferencing and meeting systems</td>
</tr>
<tr>
<td>Definition</td>
<td>Applications for supporting group decision making and for facilitating meetings despite difference in time and locations.</td>
</tr>
<tr>
<td>Sample applications and products</td>
<td>Group Systems (Nunamaker, et al. 1989); Video Conferencing Systems; Microsoft Net Meeting; WebEx; Lotus Sametime</td>
</tr>
<tr>
<td>Application Class</td>
<td>Collaborative authoring and design systems</td>
</tr>
<tr>
<td>Definition</td>
<td>Applications for group editing, shared screen work, group document or image management and document database.</td>
</tr>
<tr>
<td>Sample applications and products</td>
<td>Multi-user editing systems</td>
</tr>
<tr>
<td>Application Class</td>
<td>Collaboration and information sharing systems</td>
</tr>
<tr>
<td>Definition</td>
<td>Applications for collaboration, usually composed of an application suite, with shared work spaces and accessible or are ported through the web</td>
</tr>
<tr>
<td>Sample applications and products</td>
<td>Knowledge management systems</td>
</tr>
<tr>
<td>Application Class</td>
<td>Extensible and bundled systems</td>
</tr>
<tr>
<td>Definition</td>
<td>These are Commercial-off-The-Shelf (COTS) groupware applications, which feature an all-in-one concept by incorporating several groupware applications in one, i.e. email, discussion, video conferencing combined in one suite. Further, some of these kinds of groupware are customizable, extendable and even serve as a development environment for constructing new groupware applications.</td>
</tr>
<tr>
<td>Sample applications and products</td>
<td>Lotus Notes; Microsoft Exchange / Exchange Server ; Novell Netware</td>
</tr>
<tr>
<td>Application Class</td>
<td>Special purpose ICT with groupware aspects</td>
</tr>
<tr>
<td>Definition</td>
<td>ERP/ CRM Applications</td>
</tr>
<tr>
<td>Sample applications and products</td>
<td>Beaufort Systems (Bondarouk &amp; Sikkel 2002); PeopleSoft Lead Management</td>
</tr>
</tbody>
</table>

The listing (Table 2-4) provided above is just a sample. It is highly likely that there new groupware applications are currently under development.
2.4.6 Summary

The concept map (See Figure 2-4) is an attempt to summarize the domain of groupware and to provide an understanding of it.

![Figure 2-4: A conceptual map of groupware]

Central to this discussion is the definition of groupware as any group oriented ICT that provides support for cooperative work where in the way of working is not specified in advance. This immediately sets it apart from workflow systems which provide support for well-prescribed business processes and structured tasks that follow some rules of order and sequence.

A groupware system is a socio-technical system that is used by groups of people and is meant to provide support to different group processes. These dimensions of use and the presence of electronic facilities that support group processes are necessary components to instantiate a groupware application.

Groupware applications are used by groups of people who have shared tasks and goals. Correspondingly, people using groupware may have a higher or lower level of awareness of belonging to a group but are necessarily aware of the interdependence of their work with others, who may be distributed in space.

At the same time, groupware systems are technical systems that provide support for the following processes: communication, information sharing, coordination, cooperation and social interaction. Support for these types of processes can also be found in large composite software applications such as enterprise applications and organisation ICT. Groupware has been studied for purposes of understanding through several classification taxonomies. These taxonomies are based on several categorization schemes: application domain, time and space dimension and work content and workflow.

2.5 Requirements evolution in groupware application domains

It is generally known that introducing information technology in organization brings about changes. As a tool for mediating human interaction and communication, groupware systems have the potential to bring about changes to the social functioning of individuals, groups and
organizations. Naturally, changes in the organisation and in the functioning of groups and individuals will affect the IT requirements of the organisation. These include those which implicated the system. For groupware applications, requirements evolution is a more problematic issue for the following reasons:

- Groupware requirements are elusive to capture due to subtlety of cooperative processes; and
- When implemented, it cannot be anticipated how the groupware system will be used.

2.5.1 Groupware requirements

Groupware requirements are difficult to elicit due to the elusive nature of cooperative work. Cooperative tasks are performed both as a combination of individual and cooperative processes in which the aspect of ‘working together’ varies substantially. Therefore, they are hard to detect and capture. While the bulk of the work may be done individually and people work apart from each other, cooperation still takes place in a subtle manner. In an ethnographic study, Heath and Luff (1999) describe how employees in the London Underground informally take action to resolve crisis, i.e. one information assistant overhears conversation between line controller and train drivers about a problem, draws his conclusions and broadcasts the situation to the passengers in the underground stations. Cooperative actions such these are dynamic, informal and are executed in non-prescript manner. Aside from this, work done in other command and control centres such as air traffic control and in unpredictable environments such as stock market trading are examples of settings in which this kind of cooperative work is being done (Andriessen 2003; Hughes et al. 1996; Sommerville et al 1992, 1993).

Consequently, these tasks elude formal analysis and are hard to model in advance. While it may be possible to uncover these tasks through task analysis and observations of people’s work, it is still not possible to gain a complete understanding of the process. This is because when a tool is introduced to mediate these processes, the way in which the work is being done will change but in uncertain ways. Over twenty (20) years of research in CSCW have shown that the real requirements and utility of a groupware application can only be determined in situ (Schmidt & Bannon 1992). Or in terms of Bannon & Hughes (1993), in order to properly appreciate a groupware system's requirements, it is necessary to secure a real-world reference for its actual use in the work context.

The requirements engineering implications of these findings are (1) it is necessary to have a profound understanding of the work processes, including the tacit actions that take place, and (2) a working prototype of the system is needed to be implemented in order to improve the understanding of the requirements. To such effect, efforts have been stirred towards the utilization of theories from the social sciences and psychology towards understanding group processes and group work. At the same time, design approaches that incorporate the principles advocated by these theories are also being developed and applied in the design of groupware applications. Among these theoretical approaches meant towards understanding social processes surrounding the performance of cooperative work with and without technology mediation include activity theory (Kuuti & Arvon 1992; Bertelsen & Bodker 2002), structuration theory (Giddens 1984; Orlikowski 1992), adaptive structuration theory (DeSanctis & Poole 1990), among others. We do not intend to discuss the details of each approach; these have already been tackled in literature quite extensively.

Likewise, design methodologies which are continually being studied and applied especially in the CSCW community are given as follows.

- Ethnography (Fielding 1994; Hughes et al. 1994; Crabtree et al. 2003)
- Participatory Design (Mumford 1995)
Correspondingly, these design methodologies can be considered as requirements elicitation techniques. The differences in the use of terminologies can be attributed to the fact that scientists and researchers interested in the same phenomena are scattered in different research communities and specialization domains.

2.5.2 Groupware use cannot be anticipated in advance

Like any other technology made for human use, especially for joint use, it cannot be anticipated in advance how a groupware system will be used. There is always the difference between the planned intentions and the actions that take place in context of system use (Suchman 1986). A forerunner example of this behavior is the sms (short messaging system) functionality in mobile telephones (Hillebrand et al. 2010; Trosby 2004; Mouly & Pautet 1992). Originally intended for engineers on field to exchange messages, sms actually caught up with the younger generation of teenagers, which non-oral communication through portable devices quickly became a fad (Grinter & Eldridge 2001). Later on, the use of sms spread through other age groups and its utility in supporting communication in a cheap and more functional way than spoken communication is continually being discovered and enhanced.

With groupware, it is generally known that these applications drift when put into use (Ciborra 1996). There is a departure from the intended use of the system to the way it is going to be used in its operational environment. Ciborra (1996) cites an example of a groupware system that was implemented with the intention that it is going to be used as a workflow and personal productivity system. During the initial implementation and use of the system, the initial intentions were fulfilled; however, there was a shift towards using the application as a knowledge management system because it was the emerging management trend of the day.

Appropriation and evolving use

When successfully introduced groupware use evolves over time. Users find ways on how make use of the system by working around it to suit their tasks and their purposes. This could lead to improvisations, adaptations and unforeseen extensions to the system (Orlikowski, 1996; Karsten & Jones 1998; Pipek & Wulf 1999; Bikson & Eveland 1996; MacLean et al. 1990).

The process with which users of groupware applications make use of technology to adapt to their needs is called appropriation. According to Dourish (2003) appropriation is the process by which people adopt and adapt technologies, fitting them into their working practices. It involves customization, i.e. the explicit reconfiguration of the technology in order to suit local needs, but it might also simply involve making use of the technology for purposes beyond those for which it was originally designed, or to serve new ends. The latter is highly possible because technology is known to have affordances (Norman 1993). These are properties of technology or software systems that enable users to discover other ways to use the system other than those intended by the designer, which are made explicit through its specifications.

Appropriation is understood to be stronger and more active than adoption (Pipek 2005). Adoption simply implies that users accept the functionalities of the system and make use those that can support their tasks. Adoption can also lead into adaptation in which users change their behavior to accommodate the system. Change in behaviour could be in the form of performing new work practices due to efficiency offered by the system, i.e. faster communication and easy access to information. Appropriation therefore is a much more proactive process on the part of the users in which they make the technology appropriate to their tasks. There is an active element of construction on their part when perceiving and using the system (Bijker, 1990).

An example of how users appropriate groupware technology is studied by Pipek & Wulf (1999). They conducted a longitudinal study about the introduction, use and eventual removal of a groupware system in the State Representative Body of a Northern German State. In their study
they found out that users tried to tailor the system in accordance to their needs: first, by choosing only the functionality that is useful to them and ignoring a key functionality, which is considered to be flagship feature of the groupware application; second, one user, a secretary tried to ‘customize’ by hiding the software feature for electronic forms from other users for fear that her job will become obsolete; and, third by introducing process improvements, i.e. parallel work pattern and information dissemination vs. the sequential paper-based process. The latter was due to an increased appreciation or a wider technological frame (Bijker, 1990; Orlikowski, 1996) of the possibilities of groupware technology. The study also exemplified the presence of an expert user (MacLean et al. 1990) wherein one of the tech-savvy employees becomes an expert in customising the groupware application, discovering functionalities and eventually disseminating the knowledge and skills to other users. Specifically, this activity indicates that appropriation is also a collaborative process.

Consequently, these appropriation activities impact the requirements to the system, especially in terms of function evolution. In the same study (Pipek & Wulf 1999), the implementation of the groupware system, one the process improvements brought about the system implementation is the shortened voting and negotiating process between the body and several government states. Processing of results is reduced from 13 to 11 days. On the other hand, this advantage is feared to create a disadvantage in the sense that the number of participants in the negotiation process will be increased, which will make the decision making process more complicated. From a requirements point of view, this concern stipulates a desired software property in a way that the system should be able to accommodate increasing number of participants and must support increasingly complex decision-making processes.

**Technology acceptance and adoption**

In most instances, groupware implementation is not successful (Pumareja et al. 2003; Grudin 1988; Grudin & Palen 1995; Orlikowski 1992; Rogers 2003). Users abandon the application; they either reject its implementation or use the system for a short period and then stop using it. In other instances, the system is used sub-optimally.

The acceptance and adoption of groupware applications by its users are very important conditions in bringing about a successful implementation of the system. They bring the situation a step closer to the desired state in which the groupware system is supposed to lead to, i.e. increased productivity, improved communication quality.

The acceptance and eventual adoption of software is dependent on a lot of factors. It is argued that acceptance results from the fit between the technology and user tasks as well as a match with context of the users (Andriessen 2003; Zigurs & Buckland 1998; Orlikowski 1996; Collis et al. 1999). Failure to effectively support the regular day-to-day tasks of insurance workers led to the abandonment of a knowledge sharing system (Pumareja et al. 2005; 2003). Instead of providing assistance to users to perform their tasks better, using the system became an extra task for these users. At the same time, a non-cooperative organisational setting can stifle the potential of a Lotus Notes as a productivity tool (Orlikowski, 1996). Competitive attitudes and knowledge hoarding prevented the software from becoming a useful tool for sourcing information from fellow workers. In a loosely coupled organisation such as an educational institution, it took a year before an electronic calendaring system was adopted by the secretaries, only after a series of discussions and informal trainings (Ruël 2001). The system was implemented following a top-down approach and with little consideration given to their circumstances: moderate computers skills, concerns about RSI and strong preference to the old way of working. Finally, there are always the attributes of being human such as emotions which also play a role in how users perceive the system based on how it tries to regulate cooperative tasks such as shared editing (Ramos et al 2002, 2005).
Organizational changes surrounding groupware implementation

Implementing groupware technologies is considered to be advantageous for the following reasons (Andriessen 2003):

- It can speed up the exchange of information
- It allows easy access to information
- It allows many people to receive information at the same time
- It can increase the number of potential participants in the discussion
- It expands horizontal and diagonal contacts
- It makes it easier to reach other people

Aside from these, overcoming the limitations imposed by distance and distribution to communication is also one of the major purposes of groupware. The findings of several studies indicate the heuristics that people separated 30 meters from each other have the same low level communication as those who are separated by larger distances (Allen 1977; Kraut et al. 1990). This utility, together with abovementioned reasons, is enough for organisations to make a strategic choice in deciding to implement a groupware application.

Extensive research has been conducted on the impact of groupware on groups and of organizational organizational functioning as a whole (Galegher & Kraut, 1994; Pinsonneault & Kraemer 1993; Vogel & Nunamaker 1990). That groupware technology implementations bring about organisational change is a well researched topic area (Orlikowski 1996; Karsten & Jones 1998).

At the group level, groupware use impacts group processes and dynamics. For example, Bikson and Eveland (1996) and Connolly et al. (1996) report that the anonymity feature of a group decision support system enlarges the scope of user participation and enables the expression of negative opinions. In normal face to face meetings such are rather difficult to convey. In return, anonymity helped improve the quality of input to group decision-making. Likewise,

At the organisational level changes result from the dynamic interaction between the technical functionality of the system and the properties of the social environment in which it used. In another study, an integrated groupware system has been observed to bring about changes to the way people collaborate in the form of increased pro-activity, increased utilization and dissemination of knowledge and new forms of coordination (Orlikowski 1996). People are more predisposed to communicate given that a communication tool is at their disposal. Furthermore, natural and social barriers to communications such as distance and rank disappear.

Specifically organisational changes are effected by changes in work structure and in the decentralising impact of groupware technology. One of these changes in work structure is typified by the shift in word processing activities and or correspondence task from sectaries to managers (Pipek & Wulf 1999). With groupware, it has become possible to develop new forms work, i.e. Telework (Olson 1989; Limburg 2002), which alter and extend the notions of an organisation.

2.5.3 Requirements implications of groupware use in organizations

In the previous discussion, we have elaborated on the issues surrounding groupware requirements and its use in organisational settings. These provide the challenges for studying requirements evolution in this application domain. Specifically, there are two things that are consequential to requirements evolution from these discussions: requirements mismatch and tailoring & tailorability. These serve to inform the ensuing design implications of groupware implementation and use in organisations.
Requirements mismatch

Issues of non-adoption and non-use of groupware applications indicate severe problems relating to requirements mismatch (see section 2.5.2). This means that requirements that implicated the system do not correspond to the real needs of the users and neither does it fit the context. In other words, the design was not appropriate or not satisfactory.

Consequently, this problem is a trigger for requirements evolution, or in particular, specification evolution. The software has to be adapted to fit the real as well the emerging requirements resulting from the implementation of the system. This could mean revising software functionality by adding or improving functions, as well as enhancing quality properties, i.e. making the software more user-friendly and comprehensible.

Tailoring and tailorability

When groupware systems are intensively used, they are appropriated. Appropriation is most explicit when users begin to tailor the system – by customising it according to their purposes. Tailoring is the process of reconfiguring the software to suit user needs. Doing this does not change the product specification of the groupware. Because of this, tailorability is acknowledged to be a desirable property of groupware such that it would support appropriation.

Increasingly, design efforts have been stirred towards making groupware tailorable for the end-user. Accordingly, this calls for making groupware design more composable and adaptable to the situation. One approach to do this is by composing a groupware architecture into services (ter Hofte 1998) or in terms of shared functional layers (Dewan 1998).

2.6 Towards a conceptual framework for requirements evolution

In the previous sections, we have provided the knowledge baseline that sets the theoretical agenda for our research. Specifically, we have explored the domains of requirements, requirements engineering, groupware and the implications of groupware use on requirements. Based on what is known about these domains, we constructed an initial conceptual framework for addressing the central research question on requirements evolution for groupware applications in use.

2.6.1 Conceptual framework: tentative model

A conceptual framework is an abstraction of some aspect of reality through a structured arrangement of concepts and their possible relations. It offers a certain view for describing, explaining, diagnosing phenomenon as well as developing a plan for action (Eddins 1967). A particular phenomenon can be studied in terms of several conceptual frameworks. For example, various frameworks have been offered and used to study system implementation in organizations: improvisational model of change (Orlikowski & Hofman 1997); network orchestration view of coordination (Shaw 2007); inter-organizational strategies (Bensaou & Venkatraman 1996; Bachmann & van Witteloostuizen 2009); and guidelines for implementing mass information systems (Hansen 1995), among others.

A conceptual framework is a theoretical tool that can be used to guide and structure a research study. It helps in sharpening the focus of the investigation, i.e. what to study and it provides convenient headings under which a wide range of activity and seemingly diverse phenomena may be subsumed (Eddins 1967). For example, the socio-technical systems view of change subsumes both the social and technical dimensions of say, an ICT implementation environment, and that processes such as interaction provide opportunities for further understanding. Given that a conceptual framework is a cognitive lens from which to view phenomena and it points the
investigator on where to look, it cannot be proven right or wrong as in the case of a theory or a hypothesis (Silverman 2000). The quality of a conceptual framework can be evaluated on the basis of its usefulness. It should be theoretically adequate to provide insight into the phenomenon and practically productive in a way that it serves a source for generating hypotheses (Eddins 1967).

In line with the exploratory nature of our investigation (Sections 1.3.1 and 1.3.2), the conceptual framework developed in this chapter is a tentative model. It is an initial framework based on literature review that will be applied to data. It is updated and improved based on how well it provides meaningful and useful insights about the phenomenon of study.

2.6.2 Points of departure

The following perspectives serve as take-off points from that inform the our conceptual framework:

- Continuing process view of requirements engineering

We share the view that requirements engineering should be a continuing process (Jarke & Pohl 1994; Dobson et al. 1993; Harker 1993; Hughes et al. 1996; 2.3.3). This perspective is appropriate when seeking to understand requirements evolution for groupware applications. The continuing process view of requirements engineering finds support in the findings revealed about software maintenance activities (see 2.3).

- A continuing process view of system implementation

Our take on system implementation is based on the view that groupware implementation is a continuing design and use process. Early studies on MIS implementations regard implementation as a process that begins with the plan or intention to implement a system and remain to be a continuing process even after the system is installed and used (Gottschalk 1999; Robey & Newman 1996; Mumford 1995).

- A broad and integrated view of requirements

It is evident from the discussion in Section 2.1 that a broad and integrated view of requirements is needed. While a more software-oriented view is useful for purposes of immediately building or acquiring a software product, the underlying business and organizational rationale are equally important. This is true for applications such as groupware which require interaction between human users and whose cannot be in anticipated in advance (Section 2.5.2)

- Focus of contribution

This work describes the dynamics of requirements evolution. To make this problem more manageable, we focus on groupware systems and their organizational use. Where most organizational studies about technology implementation in organizations focus on the organizational changes brought about by technology, our work focuses on the requirements implications of these changes. We will study this by focusing on the later phases of a system's lifecycle, namely, its introduction and use in a user organization. Our study consists of an analysis of four cases of groupware implementations using the conceptual mode described next.

2.7 Requirements evolution conceptual framework: initial formulation

Our initial conceptual framework is a set of definitions consisting of two parts. Part one is a matrix of requirements domains (Figure 2-5) and part two is a model of requirements evolution (Figure 2-6).
2.7.1 Matrix of requirements domains

The requirements domain matrix (Figure 2-5) is an integration of the different interpretations of the term requirements found in literature on software acquisition, development and use (Sections 2.1.1, 2.13 through 2.1.5). It represents our consolidated view of requirements organized according to various domains of requirements: business and software requirements are taken altogether as requirements, each of which has a problem domain and a solution domain that are also considered as requirements. Business requirements consist of non-software demands as defined by stakeholders. User requirements are considered part of business requirements. Software requirements, on the other hand, are the demands ascribed to the software by the business.

![Figure 2-5. Requirements domains matrix](image)

Figure 2-5 shows to four different categories of requirements, which we will refer to as requirements domains. These domains are the **business problem**, the **business solution**, the **software product concept**, and the **software solution specification**.

To illustrate further, a requirement in the **business problem domain** would consist of the problem specifications of stakeholders where solutions are sought. The definition of a problem as the difference between what is desired and what is perceived holds true in this domain. Therefore, the business problem is the difference between the desired state of the business and its current state. The business can be used interchangeably in this case with the organisation. Any statement that specifies and provides details about this gap is a requirement in this domain. For example, an expression of an organizational concern in the form a goal describing a desired state would be a requirement representing a business problem. Specifically, business problems are concerns with an attached level of importance. This could be due to the level in which they occur in the organisation, their criticality in the smooth functioning of the business or in terms of the negative and consequences and discomfort problem owners have to bear if the problem is not solved. Business problems can be found at the organisational or at the local operational level, i.e. division or department level. In the case of the latter, task-related problems as explicated by users are also part of this domain. Ultimately, these motivate the development and/or acquisition of a software system. One way to specify a business problem is to make use of the different conceptualizations by Smith (1989) of a problem definition (See Section 2.1.4).

A **business solution** on the other hand is a non-software solution to a business problem. It could be in the form of organizational or business policies, new procedures and structures as well as business and organizational strategies. When amenable to a software solution, the business solution forms part of the requirements for the software product.

Not all business problems or business solutions are amenable to a software package realization. For those that are, the entire dimension of software requirements would comprise a solution to the business requirements. Software requirements can also be further decomposed in
terms of problem definitions and solution specifications. These take the form of software product concept and software solution specification, respectively. Each of these decompositions of software requirements can address a requirement from any of the decompositions of the business requirements, i.e. a business problem or a business solution.

In this sense, the problem definition for software requirements need not be construed as a problem with the software but is rather understood as a product concept assigned to the software by a business requirement. Thus, the term software product concept is used to refer to requirements belonging to the software problem domain. It can be further understood as a reference to the generic product type without the specifications. For example, a business solution could contain an Internet-based sales channel for a bank. This then is the product concept to be elaborated into a software solution specification.

The solution specification in the software domain on the other hand is operationalized by the software solution specification. This solution specification is but one of the many possible specifications that enacts the given software product concept. When specified, it gives an indication of what the finished product would be like. At the time of its use, the implemented software should have a product specification that corresponds to the software specification for it.

2.7.2 Requirements evolution definition

Requirements evolution is the change in requirements over time. Using the matrix of requirements domains, requirements evolution is therefore the change in the state of the requirements matrix from one period to a later period over time (Figure 2-6). When any of these domains change, the requirements are already considered to have evolved.

![Figure 2-6. A descriptive model of requirements evolution](image)

The model (Figure 2-6) is a simple descriptive model that illustrates the transition of one instance of the requirements matrix at a certain period, i.e. initial period \( p \), to a later time period \( p' \). Accordingly, this model implicates or rather defines evolution. The requirements at initial period \( p \) can represent the preliminary requirements during the time the system is being acquired or developed. In other words, the requirements justify the existence of the system. The requirements at the later time period \( p' \) represent the requirements in the later period when the software is implemented and used.

2.8 Next steps: operationalization and research design

The next step is to develop the appropriate research design that would make the conceptual framework amenable to observations and action. Further specification of the conceptual elements...
in the framework is needed such that they can be linked to relevant observable data. These steps are laid out in Chapter 3.
CHAPTER

3

METHODOLOGICAL JUSTIFICATION

This chapter is a specification of how the research will be carried out. It includes the choice of research strategy (Sections 3.1 and 3.2), design (Section 3.3) and the set of techniques and protocols (Sections 3.4 and 3.5) for gathering, analyzing and reporting data.

3.1 Case study strategy

The use of case studies as a research strategy is done in a wide range of research domains outside the scope of social science. Case studies have been widely implemented in information systems research (Myers 1997; 2010; AIS 2010), studies of ICT and organizational interaction (Carlson 1999; Shani et al. 2000), workplace studies (Heath & Luff 1992; Hughes et al. 1992; Pettersson, et al. 2002) as well as software design and architecture (Bird et al. 2009; Perry et al. 2000; Pendharkar & Rodger 2007; AIS 2010). A growing body of literature on studies of groupware implementation and use have been motivated by case studies, specifically on CSCW conferences (http://www.cscw2010.org) and the CRIWG community (http://www.criwg.org). In the domain of requirements engineering, case studies are used to provide insights about engineering practices, to determine the feasibility of a certain design approach, and to validate certain technique or to apply theory (Jiang et al. 2008; Fuentes-Fernandez et al. 2009; Molina et al. 2009). The case study strategy has been adopted to address various research purposes (Eisenhardt 1989; Yin 1994; Stake 1995; Verschuren & Doorewaard 1999; Yin 2003). These purposes include:

3.1.1 Exploration

Investigations on the impact of groupware technology on social behavior, group functioning and organizational processes (Kesner 1995; Belanger & Allport 2008) as well as inquiries into the failure and success of groupware implementations (Grudin & Palen 1995, 1997; Brown 2000; Gellatly et al. 2000; Dennis et al. 2003) can be considered as exploratory case studies. In these studies, no explicit hypotheses are offered but rather the investigation focused on research questions which bound and guide the study. Exploratory case studies can also lead to explanations (Section 3.1.3) and theory-building studies (Section 3.1.2).

A case study can be used also as a pilot study when the topic is new and little is known about the phenomenon in terms of knowledge and theories (Yin 1994; Yin 2003). For example, Hughes et. al. (1992) conducted an in-depth study of air traffic control processes using ethnographic techniques to gather observations on the interaction between the situation, workers and artifacts. The goal was two-fold: firstly, to gain an understanding how the work is done and secondly, to identify design guidelines for an appropriate support tool. While the authors did not indicate the use of case study research in their work, the use of the UK air traffic control site can be considered the case and the approach as well as the logic used is akin to case study research. It is
also an exploratory study in the sense that it did not offer a preliminary theoretical model. The absence of such a model is a key property of ethnographic studies (Stake 1983; Lincoln & Guba 1985; Yin 2003). In the light of the additional goal of developing a computer support tool, the air traffic control study can be considered as a pilot study because it provides preliminary evidence in terms of design inputs to the development of a support system.

3.1.2 Theory building

Orlikowski (1996a; 1996b) and later together with Hofman (1997) made use of an in-depth case study of a Lotus Notes implementation in a software company to develop a theory of ICT-enabled organizational change. Ngwenyama's (1998) longitudinal study of a groupware implementation is a contribution to the further development of process research theory. These studies identify changes in the communication structure, the efficacy of information exchange processes and changes to the way cooperative office work is done, thus providing inputs for the development of a theory or improvement of an existing one.

It has been pointed out that the level of generalization in theory development using case studies is at the level of theory (Yin 2003; Eisenhardt 1989; Firestone 1993). That is, the findings are used to support a broader theory. This is called analytic generalization, which is contrasted with generalizing to a population or statistical generalization. The individual case should not be construed as a sample unit (Yin 2003). Rather, it should be seen as an experiment with which the developed theory is used as a template with which to compare the empirical results (Yin 2003, p. 33).

In the instance of an emerging theory, multiple case studies can be conducted in order to confirm or further develop the theory. This is reflected in the studies conducted by Leonard-Barton (1995) on several technology companies focusing on internal technology transfer processes. The findings, which reveal lessons learned and key success factors contribute to the development of a theory of innovation.

3.1.3 Explanation

Explanation implies knowing more about a situation beyond its representation in tallies and numbers. It deals with knowing the dynamics of the events that take place, having multiple perspectives of the situation and tracing the build-up and development of events. Usually, these are addressed by questions such as how and why (Yin 2003). Case studies are used to address these questions because they allow for a comprehensive investigation and a closer examination of contextual conditions which could be relevant to the phenomenon of study. For example, a study by Dingsøyrr and Moe (2008) reports that users who participated in the creation of electronic process guides, i.e. through process workshops, used the system more, used more system functions and expressed more advantages and disadvantages than those who did not participate. Data was gathered from usage logs, direct observations, interviews and artifact inspection. In this study, qualitative and quantitative analysis of data was employed. For the similar purpose of providing explanation, Gunter (1999) made use of two comparative case studies to illustrate how companies make use of groupware technologies to support information collection and dissemination in virtual teams in order to gain competitive advantage.

3.1.4 Description and history

Stake (1995) referred to a type of intrinsic case study in which the goal is to study the case itself as given. This can be found in the documentation of Pipek and Wulf's (1999) study on the introduction, use and removal of a groupware system for a German province in support of federal and local government political processes. The study provides an idiographic description of the implementation as it unfolds. It also provided insights into individual motives and how a
groupware system can be regarded as a threat to one's job security. At the same time, Brezillon et al. (2003) also documented the idiosyncratic use of a groupware application to support the complex processes in a newspaper firm.

3.1.5 Theory- and hypothesis-testing

Case studies can be used to validate existing theories and test hypotheses. For this kind of purpose, the case study also fulfills the function of explanation. For example, the Actor Network Theory has been widely used to explain the diffusion of networked technologies (Callon 1987; Mark & Poltrock 2003; McInerney 2009; Shin 2010). Adaptive Structuration Theory (DeSanctis & Poole 1990, 1994) and the earlier theory which it is based on, Structuration Theory (Giddens 1984; Orlikowski 1992) have been used in studies of group decision support systems and groupware implementations.

It is also quite common to make use of more general theories, i.e. a meta-theory such as systems theory (Bertalanffy 1968; Beeson & Davis 2000) and socio-technical systems theory (Mumford 1985; Mumford 1995; Bikson & Eveland 1996) as points of departure in conducting the case. Later on, more specific conceptual elements such as a patterns, variables or insights are derived from the case study as a result of the application of such meta-theories. For example, in their study on the use of a conferencing system in a large and spatially distributed network organization, Mark & Poltrock (2004) confirmed the ideas of a social worlds theory (Strauss 1978; Clarke 1991).

Case studies also provide the means for more specific inquiries such as hypothesis testing. Using a comparative case study of two emerging professions in different regional settings, Montgomery and Oliver (2007) hypothesized that clear social boundaries that distinguish one's social identity contribute to the formation and sustained existence professional communities. They also made use of the case studies to validate propositions derived from a process-based model that shows how boundaries are being constructed using additional theories.

In the absence of a hypothesis, some studies make use of conceptual framework or models as theoretical starting points in conducting case studies. While this use of the case study is actually in the line of theory-building, the case study's goal is then to prove that the framework is useful. Carayannis and Turner (2006) developed a conceptual model of security technology adoption and applied this to two case studies of PKI (Public Key Infrastructure) implementation. Specifically, the model served as a tool for evaluating security technology implementation by identifying factors such as security technology characteristics and organizational task characteristics as positive contributors to adoption and organizational capability and technology complexity as stifling factors. The authors further intend to apply the model in other implementations of security technologies aside from PKI.

3.1.6 Design

Case studies are also utilized in design research (Markus et al. 2002; Hevner et al. 2004; Gregor & Jones 2007). In this type of research, the case is used as an example to demonstrate the feasibility or the usefulness of a proposed solution. In a recent publication, Karat and Karat (2010) reported their experiences in employing HCI approaches to design and evaluation in several industrial projects.

3.2 Case study as our chosen research strategy

In line with the previously mentioned research purposes, we have chosen the case study as our research strategy for the following reasons:
3.2.1 Investigation of real-life phenomenon

In order to study requirements evolution during system use, observations of actual groupware implementations in the real world are needed. System use is a contextual and emergent process, which is difficult, if not impossible, to reproduce in a laboratory setting. If ever possible, the contrived process will not represent the phenomenon properly. Gathering of primary data is needed. Furthermore, the lack of empirical studies on requirements evolution concerning software implementation and use makes it worthwhile to investigate this phenomenon as it happens in real-life. For investigations of contemporary events such as an on-going implementation, a case study is suitable because it allows direct observation of events and interviews of persons involved in the events (Yin 2003).

3.2.2 Phenomenon must take place in its natural setting

In line with the first condition, our research problem on requirements evolution implies the study of the process as it takes place. Evolution is a natural occurring process in software and system implementation. Therefore, to study requirements evolution, it has to be studied as it naturally occurs. Manipulation of subjects or events is not necessary (Benbasat et al. 1987).

3.2.3 Holistic, in-depth access to phenomenon

As discussed in Chapter 2, requirements can be formal and tacit. It is also highly contextual. For example, the requirements of one organization for groupware application may be different to another organization in a different business area, say, manufacturing vs. financial business. Therefore, in gathering observations about the phenomenon, we would like to know the ins and outs of the process and see how it unfolds from various perspectives. For this, various sources of data are needed, which in a case study are available (Yin 1994, 2003; Stake 1995; Verschuren & Doorewaard 1999).

3.2.4 Findings should contribute to a theory of requirements evolution

Our research goal as stated in the beginning was to contribute to the development of a theory of evolution. Since a case study is focused on understanding the dynamics present in a single setting, it can be used to accomplish description, theory-testing and theory-generation (Eisenhardt, 1989).

3.3 Case study design

As a comprehensive research strategy, the execution of a case study inquiry requires a plan. The plan incorporates the logic, process and components needed as well as choices to be made in order to carry out the research (Eisenhardt 1989; Yin 2003). Integrating Yin's (2003) case study design and Eisenhardt's (1989) roadmap on building theory from case studies, we designed our case study in terms of the following plan and components (Sections 3.2.1 to 3.2.4)

3.3.1 Research question and propositions

Setting the boundaries of the case study are our research questions and study propositions.

**Research question specification**

Section 1.3.3 formulates the central question of our research as how do requirements change in an evolutionary process of groupware implementation and use? This question points our inquiry towards mechanisms of change. Therefore, we can specify our research question in terms of **requirements change mechanisms in groupware implementation and use**. This focus provides us
with two useful design inputs. Firstly, we limit the scope of our investigation towards cases of
groupware applications in use (vs. those undergoing design and development). This delineation in
scope assists us in the case selection process (Section 3.3.3). Secondly, it directs our attention
towards the temporal aspects of change and in gathering evidence that establish change. It also
forces the need for identifying what has changed, which means that data gathered should aptly
represent requirements.

Study propositions

Section 2.7 provides an initial conceptual framework that offers a broad definition of
requirements, which serves as basis for defining requirements change. In order to define
requirements change and make it observable as a concept, it is useful to first break it down into
smaller concepts such as requirements and evolution.

Requirements Domains

Requirements are problem as well as solution statements of the software in question and the
business it is meant to support. There are many dimensions to the concept such that it is useful to
speak of requirements in terms of domains. This is represented terms of the requirements domain
matrix (Section 2.7.1; Figure 2-5). The definition also presupposes that a requirement can be
expressed as a statement in a requirements domain.

Requirements evolution

Requirements evolution is the change in requirements in a given situation in the course of time. Using
the definition of requirements as requirements domains, requirements change is the
observed relational difference in each requirements domain (within-domain) and the impact of
changes between domains over different periods of time (inter-domain). This is illustrated in
Figure 2-6 (2.7.2).

3.3.2 Unit of analysis

The unit of analysis (Yin 1994; 2003) whose selection is influenced by the primary research
question, defines the case study. As we are interested in requirements change mechanisms in
groupware implementation and use, our unit of analysis for an individual case is defined by an
instance of groupware implementation and use in an organization. The implementation includes
the groupware application, its sponsors and users, the context and the dynamics of how the
system is being used (Figure 3-1).

![Figure 3-1. Unit of analysis](image)

Figure 3-3 sets the boundaries of the unit of analysis to the team that is supported by the
groupware. However, we take into consideration that the team is part of a larger organization,
which for the overall organizational functioning and business goals can have an impact on the requirements and how the system is going to be used. At the same time, the groupware application can be sourced from a supplier outside the organization. We placed the supplier outside the boundary of the unit of analysis and outside the user organization to indicate that we are referring to software use, when the software is already available to its market and intended users. The primary form of interaction with the software is use and not design. It is of course possible to have a groupware application built in-house. However, whatever the implementation situation may be, it is customary nowadays that the business will have an in-house IT department responsible for managing the software resources and providing services to users. Our unit of analysis also takes into consideration the influence of these actors to the extent that they affect the use of the groupware application in question.

3.3.3 Case selection: multiple case study approach

As earlier mentioned, a case study can be compared to an experiment and the selection of cases is similar to that of an experimental design. Decisions have to be made on the sample population from which the cases will be drawn and whether a single or multiple case study is appropriate.

Theoretical sampling: use of polar examples and theoretical categories

Theory-building research relies on theoretical sampling in which the cases are chosen to fill certain theoretical considerations (Eisenhardt 1989). For example, cases are selected for the purpose of extending an emergent theory or to replicate previous findings. For these purposes, cases are selected to provide polar examples (Eisenhardt 1989), i.e. small project vs. big project, or to fill theoretical categories such as group communication, i.e. face to face, distributed same time zone and distributed different time zones. Sampling on the basis of these theoretical considerations provide the grounds for controlling variation or in terms of an experiment, manipulating the independent variables. For example, Brereton et al. (2000) conducted a case study on distributed group work based their case study on distributed software engineering processes performed by students based in three different universities. The theoretical considerations were sampled on the basis of (i) the domain of group work processes – software engineering education, and (ii) degree of distribution – inter-university project work.

For our study, we made use of the following theoretical considerations in selecting the appropriate cases for our research: type of use and groupware application architecture

Polar examples: type of use - continuing use vs. about to be discontinued use

Given our focus on groupware implementation and use, we would like to sample cases that would demonstrate two extreme situations. We have identified these two situations to be:

- continuing use of groupware, which we would construe as successful implementation
- about to be discontinued use of groupware, which, on the other hand is regarded as failed implementation

Groupware application architecture: random

Section 2.4.4 concludes that any ICT application that supports group processes can be considered groupware. This way, any software application in use that fits this criteria is a candidate sample for our study. What is more important is that cooperative work processes are present in the case. In other words, it is the group that defines the software as groupware.
Multiple case study design: 2x2 matrix

For our purpose of developing a theory of requirements evolution, the use of multiple case studies is appropriate. Based on the theoretical sampling criteria we have set, our design calls for a 2x2 case study matrix (Figure 3-2) for which 4 case study projects are needed.

![2x2 matrix case study design](image)

Figure 3-2. 2x2 matrix case study design

In this multiple case study design (Figure 3-2), we try to incorporate the strategy of polar examples by finding cases with continuing and discontinued groupware use. Likewise, for the purpose of extending an emergent theory, we would like to have comparable cases based on the type of groupware use criteria. Therefore, we would like to have 2 cases of continuing use of groupware and 2 cases of discontinued use of groupware.

3.3.4 Data collection

The data collection process is specified in terms of the methods and the protocol used in the case studies.

**Methods**

One of the advantages of the case studies is the variety of methods that can be used in gathering data. As a holistic research strategy aimed at capturing various perspectives and the richness of context, case studies rely on multiple sources of evidence such as observations, interviews, document and archival analysis, all of which could either be qualitative or quantitative in nature (Stake 1995; Remenyi et al. 1998; Yin 2003). Table 3-1 lists the methods that best suit our data needs for each respective research item of interest that we like to know more about.

**Protocol**

In the following tables and questions, we have worked out our research question and study propositions in smaller conceptual units amenable to observation. In addition, we also incorporate how we plan to gather data on these items.

**Requirements domains**

Table 3-1 is a specification of the different requirements domains, the sources of observation and the methods that can be used to gather representative data. Additional explanation is also provided for each domain specification.
Table 3-1: Requirements domain specifications for data gathering

<table>
<thead>
<tr>
<th>Requirements Domains</th>
<th>Specification</th>
<th>Source and methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business problem</td>
<td>The goals as to why the groupware was implemented.</td>
<td>Interviews with system sponsors and users</td>
</tr>
<tr>
<td></td>
<td>Statement of issues and difficulties encountered in using the groupware.</td>
<td>Project plan and documentation</td>
</tr>
<tr>
<td></td>
<td>Data representing business problem domain: goal specification from stakeholders involved; breakdowns in system use</td>
<td>Observations</td>
</tr>
<tr>
<td>Business solution</td>
<td>Non-software approaches aimed at addressing the business problems within the scope of the implementation</td>
<td>Interviews with system sponsors and users</td>
</tr>
<tr>
<td></td>
<td>Proposed changes in the way how people work, how the business is to be carried out or how the software is to be used in order to improve a problem situation.</td>
<td>Project plan and documentation</td>
</tr>
<tr>
<td></td>
<td>Data representing business solution: new business strategy, policies (formal and informal), agreements, suggestions for improvement, process or work re-design, re-organization</td>
<td>Observations</td>
</tr>
<tr>
<td>Software product concept</td>
<td>The generic product idea without the specifications</td>
<td>Interviews with system sponsors and users</td>
</tr>
<tr>
<td></td>
<td>Mental model or metaphor assigned by users to the system</td>
<td>Project plan and documentation</td>
</tr>
<tr>
<td></td>
<td>Data representing software product concept: software product packaging; articulations of users' mental models</td>
<td>Observations, Artifact inspection, Software product website</td>
</tr>
<tr>
<td>Software solution specification</td>
<td>System description on the basis of its functional properties, quality attributes, data and operating requirements and constraints</td>
<td>Interviews with system sponsors and users</td>
</tr>
<tr>
<td></td>
<td>The product specifications of the implemented groupware application.</td>
<td>Project plan and documentation</td>
</tr>
<tr>
<td></td>
<td>Data representing software solution specification: software specification document, functional specification, quality requirements, system requirements and constraints</td>
<td>Observations, Artifact inspection, Software product website</td>
</tr>
</tbody>
</table>

To gather data about the **business problem domain** is to identify the goals stakeholders have in implementing the groupware application. Therefore, the goal of the data gathering process is to uncover the reasons why the groupware application was implemented and being used. The goals are not only from the business, the organization or the system sponsor as a whole. It also include user goals. For the individual user, it is interesting to know what are his or her goals in using the groupware. Why is the software being used or alternatively, not being used? Our approach to gathering data about this domain includes interviews with relevant stakeholders – system sponsors and end-users, a review of project plan and documentation and observations.

The **business solution domain** on the other hand is represented by non-software approaches
such as business strategy, re-organization, policies both formal and informal, agreements, suggestions for improvement, work or task re-design, decisions to be made and contemplated actions aimed introducing an improvement to the problem situation. It is important to remember that non-software does not also mean the absence of software. It could be that the problem situation is about the unsatisfactory and minimal use of an implemented ICT application. However, for this domain, we limit the specification of solutions to statements that do not contain specification of software properties. Measures, in terms of business solutions, can be taken and proposed such as introducing training in order to further stimulate software use. Just like the business problem domain, the scope of the business solution domain is defined by the groupware implementation. The methods to employ gathering data also include interviews, review of project documentation and observations.

The **software product concept** is the generic product concept without the software specifications. We noted in Chapter 2 that this is closely associated with how the software is packaged and marketed to its users. It is the software product idea that is evoked when interacting with the software. To translate this into observable phenomenon, we also specified this domain as the mental models or metaphors assigned by users to the system. Through interviews, we can ask users and system sponsors to articulate their mental models and ideas. A review of project documentation also provide clues on how they expressed and worked out their mental models. Direct interaction with the software, a review of the product website, if available, and observation also provide the researcher another view of the software product independent of the users.

Finally, the **software solution specification domain** is the description of the desired or implemented system in terms of functional, quality, data and operating requirements. In an on-going groupware implementation, the software solution specifications are the product specifications. A primary source of data for this domain, if available, is the software requirements specification document. The specifications can also be derived through artifact inspection, i.e. interacting with the software, project documentation, software product website and observations.

**Requirements evolution**

Section 3.3.1 specifies requirements change as the change in any of the requirements domain over time. Therefore, data to be gathered on the different requirements domains should also reflect the different time periods. To go about identifying the different periods, the temporal scheme (Section 3.4.1) is used. Using the implementation phases scheme as indicators of periods, requirements change can be tracked by identifying requirements at different implementation phases (Table 3-2). Ultimately, the goal is to be able to make comparisons between the data from the different periods and draw conclusions about the change that took place.

**Table 3-2: Data gathering protocol for requirements change**

<table>
<thead>
<tr>
<th>Requirements Domains</th>
<th>Temporal Schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business problem</td>
<td>Pre-Implementation</td>
</tr>
<tr>
<td>Business solution</td>
<td>Early Implementation</td>
</tr>
<tr>
<td>Software product concept</td>
<td>Post Deployment</td>
</tr>
<tr>
<td>Software solution specification</td>
<td></td>
</tr>
</tbody>
</table>

**Levels of questions**

Key to the implementation of the data collection protocol and the generation of data is a hierarchy of questions that addresses the different concerns in the study. For example, there is a difference between the questions asked of the case and the interview questions that will be asked on the field
from the case study subjects. Likewise, there is also a difference in an inquiry made in a single case and in multiple case studies. In case of the latter, one is interested in finding out commonalities and patterns. For this purpose, Yin (2003; p. 74) have identified 5 levels of questions:

- **Level 1: interviewee questions**
  These are the lowest and most particular types of questions.

- **Level 2: individual case questions**
  These are questions asked of an individual case, which are different in content and intention than that of questions asked of interviews. The individual case may be a part of series of case studies.

- **Level 3: multiple case study questions**
  These are questions towards finding patterns across multiple case studies. This is only relevant to multiple case study designs.

- **Level 4: research questions**
  These are questions asked of an entire study and may very well correspond to the research questions.

- **Level 5: normative questions**
  These are questions that go beyond the scope of the study. Asking normative questions leads to policy recommendations and conclusions.

Among these levels of questions, Level 4 has already been addressed. For data gathering purposes, Levels 1 to 3 are relevant. Level 5 are questions addressed at the conclusion of the research (see Chapter 10).

**Level 1: Interviewee questions**

Some of the interviewee questions are derived direction from Level 2. See Appendix A for these questions.

**Level 2: Questions for the individual cases**

These are the questions we would like to have answered at the end of one case study. These questions extend and transform the research questions and propositions into data generating inquiries.

L2Q1. What are the requirements (broadly defined in terms of our proposed requirements domains matrix) that motivate the groupware implementation in this case?

L2Q2. Who are the relevant stakeholders?

L2Q3. What is the groupware application in use?

L2Q4. What has changed in the requirements (based on L2Q1) ever since the groupware was implemented and how do the changes affect the requirements domains?

L2Q5. What are the factors that contribute to the change (follow-up question from L2Q4)?

L2Q6. What emerging patterns arise from the case about these changes?

L2Q7. What lessons can be learned from this groupware implementation either as a failed or successful implementation?

L2Q8. What are the implications of the findings from the case to our conceptual framework?
Level 3: Pattern-seeking questions across multiple cases

Some questions asked of the individual case are also applicable at the level of multiple case studies. Specifically, this is relevant to questions that pertain to patterns of changes and theoretical implications.

L3Q1. What are the common patterns of change among the cases?

L3Q2. What can be learned from the aggregated findings across cases about requirements evolution as a phenomenon?

L3Q3. What can be concluded about the conceptual framework and its application in a multiple case study research?

Level 5: Normative questions about policy recommendations and conclusions

These questions are an attempt to extend the practical usefulness of our research. On the basis of the conclusions and lessons learned, it is possible to identify several recommendations for changes and action. These questions will be the addressed in the final chapter of this thesis.

L5Q1. On the basis of the study conclusions, what actions or guidelines can be given to organizations hosting groupware that will help them manage the process more efficiently and purposefully?

L5Q2. What advice or heuristics can be given to designers of collaboration technologies such that the uptake and use of these technologies are indeed supportive of human activities and purposes?

3.4 Data analysis protocol

According to Yin (1994, 2003), analysis of case study evidence is one of least developed and most difficult areas of case study research. Various strategies have been suggested and we will make use of two approaches: using a temporal scheme and relying on theoretical propositions.

3.4.1 Temporal scheme

A chronological order or a temporal scheme is useful way to present qualitative research data (Miles and Huberman 1994). For our purposes given that we would like to monitor changes in requirements over time, a temporal scheme is desirable.

Our main analytical activity is reconstructing the events leading to and during the implementation, and derive the relevant requirements. We will make use of implementation phases as temporal schemes in reconstructing the events. Implementation phases are temporal sequences that comprise an implementation process. A phase is an interval in the implementation process around significant events that can characterized with a common theme. In terms of Miles and Huberman (1994), a phase represents an episode of nuance. Through these phases, we can track and monitor changes to requirements because we can distinguish one phase from the next one. We distinguish a sequence of three phases: beginning (pre-implementation), middle (early implementation) and end (post deployment).

Pre-implementation is the period immediately preceding the groupware implementation.

Early implementation is the period around the official introduction of the groupware in the organization. It is characterized as a period of adjustment in which users make the transition from an old way of working or an old system to a new one with the groupware.

Post-deployment is the phase in which system use is generally agreed to be beyond the transition phase. The software is no longer considered to be new. Other purposes for the software, whether
intended as per design, are being considered and proposed

Each implementation project is unique and particular aspects of a project can vary from one organization to another. Therefore, there is no fixed amount of time that approximates how long a period is. The interval as to how long a phase lasts is rather flexible.

3.4.2 Relying on theoretical propositions

The theoretical propositions (Section 3.3.1) in combination with the temporal scheme (Section 3.4.1) are used as a structure to gather and analyze data from the cases. The analysis features two main categories: requirements and requirements change.

Requirements: requirements domains as data categories

The requirements domains – business problems, business solutions, software product concept and software solution specification are used as categories for categorizing data on requirements.

Requirements evolution

The following techniques are used in structuring evidence indicating requirements change:

- Within-domain monitoring of requirements at different phases: what changes took place in each of the four domains?
- Identifying impacts across domains at different phases: what are the impacts of changed requirements in other domains?
- Seeking patterns in the changes and identifying relationships in the data.

3.5 Challenges in data collection

As a method for observing a rich representation of reality which subsumes the phenomenon of interest, gathering data from case studies can pose some challenges. Firstly, data from one case alone can be arguably infinite, as evidence can come from multiple sources. Secondly, in a study of change such as this, the need for case materials extending over longer periods of time introduces some risks to data quality and threats to validity.

The first challenge is a known issue in qualitative data gathering. Stake (1995) identifies budget and researcher attention as practical issues that limit the scope of what can be observed and therefore of the data that can be gathered. With a limited scope, comes the issue of the unknown – whether the data that has been gathered are useful and appropriate in contrast to what has not been gathered. A way around this issue accordingly is to go back to the conceptual framework and research questions and focus the data gathering efforts around these (Stake 1995). It limits the scope, but also it increases the relevance of the data gathered. Through the formulation of our research framework (Section 3.3) and its decomposition into more specific concepts and questions (Section 3.3.4), we sought to work around this initial practical issue when we chose the case study methodology. The data gathering processes help us to look for specific data that represent concepts such as business problems, business solutions, software product concept and software solution specifications. This, in the field, translates to specific data gathering methods such as interviewing people, gathering documentation, observations in the office setting and observing the software artifact itself. Applying these methods also enable triangulation (Denzin 1978) which helps increase confidence in data quality.

Secondly, a case study about long-term software implementation projects requires that data be gathered at various periods in the implementation. For example, data gathered should be able to
cover various phases such as pre-implementation, early implementation and post-deployment. Documents may exist from each of the phases, however it may happen that not all documents from the past can be retrieved. More seriously, in the ideal case, observations and interviews must take place at different moments in time, in order to cover the evolution of the system’s requirements first-hand. However, also this situation is not completely risk free. One cannot determine early on in the implementation if the project will turn out be successful or not. It is possible to ask interviewees about their thoughts sometime in the past, but this introduces the risk that they might remember things wrongly or with a certain bias, colored by hindsight. The alternative would be to limit the time span of the case studies to the period that can be observed first-hand in which the outcome is already known, i.e. whether the implementation is challenged or successful. From such a starting point, it is possible to reconstruct the events that took place and have led to the outcome. This approach would improve the quality of the data, but would reduce the time span that has to be considered.

In the light of the nature of our topic of interest, which is requirements evolution, we chose the case studies with a history, those whose implementation already exists for some time. This introduces challenges to data quality but it also has its benefits. For example, challenges such as respondent maturity are no longer relevant. The analysis and reporting of data take a more historical reconstruction approach in which the available sources of data that can be recovered are used. The consequence of this approach is also that we can draw only conclusions that would remain the same under minor changes in the data. If we find a pattern of evolution that exists only in a single instance, it might be the case that it is based on poor quality data, rather than actual events happening at the time. However, the discovery of a pattern that occurs in multiple instances across cases gives evidence that the pattern reflects reality.

3.6 Case write-up

The following is a general outline for reporting the cases (Table 3-3). The outline is aligned to the data analysis protocol and identifies the main parts of the report.

*Table 3-3. Case report outline*

| Introduction
| Case Background
| · Implementation setting
| · Organizational setting
| · Stakeholders
| · Methods
| Analysis
| Requirements
| · Requirements at pre-implementation
| · Requirements at early implementation
| · Requirements at post-deployment
| Requirements evolution
| · Within-domain requirements evolution
| · Inter-domain requirements evolution
| · Emerging patterns of change
| Discussion |
CHAPTER 4

CASE STUDY A: KENNISNET@ ACTIVE INSURANCE GROUP

Enthusiastic participation but system is not used

In this chapter, we present the first case study for this research. It is about a failed implementation of a knowledge management system in a large insurance company. A key distinguishing feature of this case study is the enthusiastic participation of the users in the design process. Nevertheless, the implementation ended up in a discontinued use of the system.

The conceptual framework (Sections 2.7 and 3.3.1) and its implementation into data collection and analysis protocols (Sections 3.3.4 and 3.4) is applied in this pilot study. The goal is to learn about the dynamics of requirements change using the conceptual framework as an analytical tool. At the same time, we would like to find how useful the conceptual framework in achieving the goal and what improvements are needed. The findings are reported in the following sequence. The case background covering the organizational set-up, implementation setting, stakeholders, methods and protocol are discussed on Section 4.1. Sections 4.2 through 4.4 are ex-post reconstructions of the case using the temporal scheme (Section 3.4.1) out of which requirements statements are derived. The analysis on requirements evolution is covered in sections 4.5 and 4.6. Section 4.7 concludes this chapter with a discussion of the case study questions and propositions, scope and lessons learned.

4.1 Case background

Active Insurance Group (AIG) is one of the largest insurance companies in the Netherlands, with approximately 12,500 employees all over the country. It specializes in the domain of financial services and all types of insurance products. The growth and evolution of AIG as a big holding company is a result of 10 years of mergers and acquisitions of more than 30 companies. AIG bought small and specialized independent insurance companies and merged with the larger and more diversified ones. Prior to these mergers and acquisitions, the individual companies that currently make up AIG used to be competitors. Before the merge, the single largest independent insurance company in the current AIG structure was a key and dominant player in the insurance market. In fact, the business campaign for mergers and expansions was initiated by that company.

The buy-outs and mergers still continue up to the present day, which is why the whole organization is faced with the challenge of unifying merging processes. It is a complex process given that the former individual companies each have their own product lines, processes, people and culture. The idea that the employees of each subsidiary have to realize that they now belong
to one company is a major organizational concern. The current organizational structure of AIG is in a dynamic state of flux so that any structural representations are bound to change over a short period of time.

AIG is a distributed organization. It has different offices in various locations within the Netherlands. The logic of the dispersion is still based on the previous manner of operations: the former sub-companies retain their offices and operations autonomously at the same location. However efforts are being undertaken to share projects such that products and services are no longer identified through the sub-company but that of AIG in general.

4.1.1 Implementation setting

The introduction of KENNISNET at AIG was a localized small-scale implementation of groupware technology. It was implemented at the non-life insurance division, which is one of the business units that make up AIG. A division represents a major product line or service. Each division is made up of the formerly independent small companies that were bought by AIG. These are referred to as brands. For example, the division Care is made up of seven different sub-companies, or brands.

Championing the implementation is the knowledge center which supports the non-life insurance division. KENNISNET is one of the many projects of the Knowledge Center Schade (KCS) to stimulate knowledge sharing among insurance experts consisting of product managers and actuaries.

4.1.2 Stakeholders

There are two groups of stakeholders directly involved in this case as system sponsors and users.

System sponsors: Knowledge Center Damage Claims (KCS)

The Knowledge Center Damage Claims (Kenniscentrum Schade, KCS for short) is the competence center for the non-life insurance division. Its existence as a unit within AIG was a result of company-wide re-organization. The center functions as a strategy support unit because it is responsible for executing the strategic policies of the organization such as knowledge management. It is made up of 5 people: manager (1), actuaries or insurance mathematicians (2) and project leader (1) for the KENNISNET project and an internal consultant from the mother company. Eventually, the software developer who developed KENNISNET became part of this group as a system proponent. The main function is to manage a loose collection of 33 insurance experts from the non-life insurance division consisting of product managers and actuaries. These insurance experts would comprise the primary target user group of KENNISNET.

Altogether, the responsibility of KCS to these experts is to develop professional competence by facilitating the creation, development and maintenance of non-life insurance knowledge. In addition, there is also a more strategic, background goal aimed at community building. The KCS Manager explains,

I feel responsible for building a team out of all non-life insurance specialists in AIG. They all work as product developers and actuaries. Two to three years ago, they all felt themselves more as competitors than colleagues. Step by step, it develops better. They work in different places but I believe they now feel together.

To realize these responsibilities, the KCS embarked on a long-term knowledge management strategy consisting of:

• Personal team-building approach through regular face-to-face workshops and meetings; and

• Implementation of a virtual, IT-based knowledge sharing support system
The regular face-to-face workshops were aimed at facilitating the experts to get to know each other and to bring them together to share experiences and expertise. Apart from these workshops, monthly knowledge management meetings were also held among the representatives of each subcompany, more commonly referred to as business units. The virtual IT-based knowledge sharing system is aimed at supporting the exchange of knowledge among the experts. It is meant to be the central place in which knowledge can be stored and accessed given that the experts are distributed in different locations. KENNISNET is the realization of this part of KCS’s KM strategy.

**System users: product managers and actuaries**

The primary target end-users of the system are the non-life insurance experts in the non-life insurance division of AIG. These individuals consist of product managers and actuaries who are located at various offices in the Netherlands. Although the insurance professionals now work for a single, spatially distributed division of AIG, in the current set-up they still formally report to the business units to which they belong – the former companies, which still operate autonomously even after the merger.

Product managers are responsible for developing and managing non-life insurance products. This is done by conducting market research, competitor analysis, knowledge and information monitoring about changes in legislation, new related products developments, and keeping track of all product development efforts within the whole company. There are three categories of non-life insurance products: (i) mobility -- private cars, motorbikes, caravans, lorries, etc., (ii) recreation -- boats, yachts, travel, etc.), and (iii) home insurance -- valuables, legal services, glass, fire damage, third party liability, etc.

Actuaries are mainly responsible for the calculations and financial analysis of insurance premiums, benefits, reserves and other number-crunching activities required in coming up with a new insurance product or policy, as well as in re-packaging an existing product or policy.

### 4.1.3 Methods and protocols

The case study was conducted for a period of 6 months and was started 7 months after KENNISNET was implemented. Data gathering was done mainly through qualitative means – interviews, observations, document analysis, and software artefact inspection. Twenty-five in-depth interviews were conducted with the various system proponents, system designer and end-users. Each interview lasted for approximately an hour to an hour and half. The number of interviews and the data gathering methods used are shown in Table 4-1.

*Table 4-1. Case A. Data collection methods and number of interviews*

<table>
<thead>
<tr>
<th>Methods</th>
<th>Source(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interviews</td>
<td>Knowledge Center Schade (KCS) – 6 out 6 members, system developer included were interviewed</td>
</tr>
<tr>
<td></td>
<td>Product Managers – 12 out of 33 were interviewed</td>
</tr>
<tr>
<td></td>
<td>Actuaries – 7 out of 11 were interviewed</td>
</tr>
<tr>
<td>Document Analysis</td>
<td>KENNISNET Project Plan</td>
</tr>
<tr>
<td>Artifact inspection</td>
<td>KENNISNET application</td>
</tr>
<tr>
<td>Observation</td>
<td>Work setting at different locations</td>
</tr>
</tbody>
</table>

Guiding the data collection process are the set of questions formulated in Chapter 3 (3.3.4 Levels of Questions). Specifically, we applied the questions at Level 1: Interviewee questions and Level 2: Questions for the individual questions. The interview questions are shown in Appendix A.
4.2 Requirements at pre-implementation: from strategy to technology

The initial idea for setting up a system for knowledge exchange started a year before the KCS was formed. The idea had actually been floating around since the KCS was founded. It had its origins from the top management strategy of implementing knowledge management within AIG. As mentioned in Section 4.3.2, unifying the merging processes resulting from years of fusion was a major organizational concern. Establishing a community feeling among the employees eventually became an organizational goal. In this light, adopting a knowledge management strategy was considered to be a means for bringing together all the employees from the different sub-companies, to evoke in them a sense of community and belonging to ‘one AIG’ and thereby achieving the organizational goal. This strategy was realized through the creation of different knowledge centres for each division, of which KCS is one. These were given the task of discharging the knowledge management strategy of the organization.

Several members of KCS were very keen on the concept of knowledge management, most especially on the idea of a software system to support it. It was the KCS Manager who proposed to establish a virtual knowledge exchange platform for the non-life insurance experts. This idea was met with enthusiasm by the rest of the KCS group. They contacted the IT department of AIG with whom the task of developing a system was requested.

In the process of developing the first version of the system, KCS faced one major constraint – they were lacking budget. They could not allocate much resources to the project, as it was a local unit initiative. Hence they were constrained to make use of existing resources, like Lotus Notes, which was already being used in AIG. The design decision therefore led to the development of a system for knowledge exchange using Lotus Notes. This was thought to be an optimal platform for launching a knowledge sharing system, given that it was already available to everyone in the organization and that people already know how to use it.

On the other hand, there were some objections to the use of Lotus Notes as the development environment for KENNISNET. According to one member of the KCS:

At the beginning, we were constrained to use Lotus Notes. At that time, we had a software engineer in our group. He was quite negative about it. He said if we are to use Lotus Notes, we better stop it. It does not have lots of possibilities. It [KENNISNET] cannot be web-based because it is not supported [by Lotus Notes]. But we thought, it is better to have something than nothing.

Subsequently, the development of KENNISNET proceeded with hardly a set of requirements. One of the member of the KCS stated the following:

In the beginning it was all very vague. We had this idea and we want to implement it.

The initial version of KENNISNET consisted simply of a database shell where information can be entered and a question and answer module. It was put into use for a limited period of time, and for a limited group of users, mainly to input information. The KCS was not satisfied with it and this triggered lots of discussion within the group. The desire to improve KENNISNET led KCS to include the non-life insurance experts in the re-design process.

4.2.1 Codes and notations

In order to uniquely identify each requirements statement, we will make use of abbreviated codes. Codes will be used to identify the phase and domain in which the statement belongs (Table 4-2). A requirements statement belonging to a certain period and domain will be numbered and will be incremented accordingly. The code will have the following notation <Implementation Phase>_<Requirements Domain>_number. For example, the very first requirements statement
from the early implementation phase belonging to the business solution domain will be coded as follows: EARLY_BS_1.

Table 4-2. Codes and notations for requirements statements

<table>
<thead>
<tr>
<th>Classification</th>
<th>Codes and Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation Phase</td>
<td></td>
</tr>
<tr>
<td>PRE</td>
<td>Pre-Implementation</td>
</tr>
<tr>
<td>EARLY</td>
<td>Early Implementation</td>
</tr>
<tr>
<td>POST</td>
<td>Post Deployment</td>
</tr>
<tr>
<td>Requirements Domains</td>
<td></td>
</tr>
<tr>
<td>BP</td>
<td>Business Problem</td>
</tr>
<tr>
<td>BS</td>
<td>Business Solution</td>
</tr>
<tr>
<td>SP</td>
<td>Software Product Concept</td>
</tr>
<tr>
<td>SS_F</td>
<td>Software Solution Specification – Functional Specifications</td>
</tr>
<tr>
<td>SS_Q</td>
<td>Software Solution Specification – Quality Specifications</td>
</tr>
<tr>
<td>SS_O</td>
<td>Software Solution Specification – Others</td>
</tr>
</tbody>
</table>

4.2.2 Requirements Statements

An overview of the requirements statement derived in this period is summarized in Table 4-3. The requirements cover the different domains of requirements as proposed by the conceptual framework. What is also notable about the requirements is that the requirements belonging to the business domains, i.e. business problem and business solution, are more or less explicit, and they contain additional information. However, the requirements pertaining to the software domain, especially the software solution specification are vague and incomplete.

Table 4-3. Case A. Requirements at Pre-Implementation

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BP_1</td>
<td>Active Insurance Group intends to respond to continuing mergers and acquisitions.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BP_2</td>
<td>One of the goals of the organization is to build a community of employees and to evoke a feeling of ‘one AIG company’</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BP_3</td>
<td>The KCS - non life insurance aims to translate the organizational goal of integrating and unifying the different sub-companies by developing a community of non-life insurance experts</td>
</tr>
<tr>
<td>4</td>
<td>PRE_BP_4</td>
<td>The KCS - non-life insurance is tasked with developing the competency of the non-life insurance experts</td>
</tr>
<tr>
<td>5</td>
<td>PRE_BP_5</td>
<td>The non-life insurance experts would like to perform their tasks efficiently and to be able to gather the information they need from colleagues in other division in an uncomplicated manner</td>
</tr>
<tr>
<td>6</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building</td>
</tr>
<tr>
<td>7</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
</tr>
<tr>
<td>8</td>
<td>PRE_SS_F_1</td>
<td>The system should store and remember data</td>
</tr>
<tr>
<td>9</td>
<td>PRE_SS_O_2</td>
<td>The system should be developed in Lotus Notes</td>
</tr>
</tbody>
</table>

PRE_BP_1. “Active Insurance Group intends to respond to organizational challenges brought about by on-going mergers and acquisitions.” This is a requirement instantiating the business problem domain. It describes the underlying phenomenon and context that motivated the creation of KENNISNET. We discovered this requirement from the articulations of system proponents, namely KCS members who refer to this statement as an top-level justification of their decision to implement a knowledge management system.

The continuing mergers and acquisitions present AIG with complex organizational changes arising from the differences in how management and business are carried out in each sub-
company. For example, all formerly independent sub-companies continue to operate under their original business labels. However, this is inefficient and not favourable because processes were being duplicated in other sub-companies and the particular strengths of each sub-company were not being optimally harnessed for improving AIG’s market competitiveness. In turn, the organization would like to respond to these challenges brought about by its growth and expansion.

PRE_BP_2. “One of the goals of the organization is to build a community of employees and to evoke a feeling of ‘one AIG company.’” This is another requirement that is more or less expressed as an explicit organizational goal. A representative of the holding company who consults for the KCS for their KM activities had confirmed it. Unifying and merging is a complex process for AIG given that the former individual companies each have their own product lines, processes, people and culture. The idea that the employees of each subsidiary have to realize that they now belong to one company has become a major organizational concern. Therefore, the business problem as a requirement is manifested in the form of a need to integrate and harmonize the processes and people of AIG. The goal is to create an organization where all employees feel that they belong to one organization despite working locally in a subsidiary in another location.

PRE_BS_1. “The adoption of a Knowledge Management (KM) strategy is a means of community-building.” This is a requirement that can be classified in the business solution domain. It is a specification of one possible solution to a business problem. In this particular case, this solution is a specific choice of action in the form of a business strategy which is aimed at addressing several business problems, namely the requirements PRE_BP_1 and PRE_BP_2.

PRE_SP_1. “The system should be a knowledge management system for enabling knowledge exchange.” This is a requirement that is made apparent from the interviews and from the interviewees’ account of the beginnings of KENNISNET. It is a requirement in the software product idea domain because it indicates the kind of software product the proponents would like to have. It gives an indication of what the software is generally about, i.e. the subject domain of the product. Proceeding from the organizational knowledge management strategy of the organization, KCS seized the opportunity to take a step further in implementing the KM policy at the division level by implementing an ICT tool support. Basically, the software application that KCS had in mind was a knowledge management system mainly for enabling knowledge exchange. The development and implementation of this application forms part of their KCS’ goals (see Section 4.3.3).

PRE_SS_F_1. “The system should store and remember data.” This is a requirement that is also derived from the interviews with users. It is a very vague requirement in the sense that details concerning the database were not clearly specified. What kind data the system will store, what will be the structure of data and how it will be stored and accessed were not specified. In fact, what the KCS actually had in mind when they mentioned database was a system containing data, or in other words a repository. Therefore, the real requirement is that the system should have data. This requirement is classified as a solution specification in the sense that it suggests a more or less identifiable implementation of a solution. It is a functional specification.

PRE_SS_C_2. “The system should be developed in Lotus Notes.” This is a requirement that can be considered more as a design constraint. However, it is very precise formulation of a design specification by identifying in which environment the system is to be developed. This way, it clearly falls into solution specification domain. The choice of Lotus Notes as a development platform for KENNISNET springs from the lack of budget for system acquisition and development from the part of KCS.

The stakeholders, their problem definitions and individual goals also form part of the business problem domain. In the defining stakeholder problems, we will include top management as part of the stakeholders. This is because they represent the interests of the organization which underlie
the motivations for implementing the system.

PRE_BP_3. “Top Management: Recognising the need for harmonious fusion, the goal emanating from the top management is to establish a notion of one-community, one-company within AIG.” This goal specified in this requirement is encompassed by the concern towards the growing organization, resulting mainly from the strategic actions that they take. For example, buying or merging with another company is largely a top management issue.

PRE_BP_4. “Knowledge Center-Schade (KCS): The problem of the KCS is the translation at the division level of the goal specified by top management.” This is realized in terms of the following two goals: (i) to develop a community of non-life insurance workers and (ii) to develop their competence. This is a requirement in the business problem domain, which is an adapted version of BP2 to the division level. This way, it is much more specific definition of a problem.

PRE_BP_5. “Non-Life Insurance Experts: The problem besetting the non-life insurance experts is focused at the task level.” Their concern is to perform their tasks efficiently, and gather the necessary information from their colleagues in the division in an uncomplicated manner. This requirement can be further specified in terms of cooperation and information sharing among each other being considered to be a problem: employees maintain the impression that the subsidiaries which they belong to are still competing with each other. On the other hand, if ever there were information or knowledge exchanges that took place, it was distributed without an awareness of competition. Likewise, information gathering activities were often duplicated at five locations. Employees were not familiar with each other’s expertise and experience.

4.2.3 Remarks

The vagueness and lack of completeness in the specification of software solution can be attributable to the informal processes in which the organization sought to generate the requirements. For one, the requirements originated from only one stakeholder group, who are not the actual end-users of the system. Second, the manner in which the requirements were derived is purely ad-hoc. Lastly, there was virtually no technical documentation that was produced to somehow serve as a design reference to guide the development of the system. The requirements given above are reconstructed and restructured from documents and interviews of what appears to be software requirements specifications for the system.

4.3 Requirements at early implementation: involving end-users in system design

KCS was not satisfied with the initial version of KENNISNET. This triggered a lot of discussion within KCS. Eventually it was proposed that the end-users – the non-life insurance experts, be included in the redesign of KENNISNET. This marked the second design iteration as well as the beginning of a participatory design process for KENNISNET.

To carry out the plans for redesigning KENNISNET, the KCS Manager met with several representative experts from the non-life insurance division. These were asked whether a knowledge-sharing system to support their knowledge exchange processes is something that they needed. Product managers indicated they need such a kind of system because they would like to search for information about competitors and products, which other business units have information about. In general, the experts felt the need for such kind of system and therefore, the idea to redesign KENNISNET was also met with enthusiasm.

The second design iteration for improving KENNISNET began a new project structure and an expanded set of system sponsors. A new member joined KCS a few months earlier and assumed the project leadership of KENNISNET. This project leader together with a select group of experts plus a software engineer from the IT department formed the design team. The group of experts were representatives from the five business units that make up the non-life insurance division.
The redesign process was set into motion through a series of design workshops. In these workshops, the participants threshed out their requirements. Business and project plans were drafted, with more specific but ambitious goals for the project. Specific requirements were elicited in the form of functional demands. One particular output of the design workshops, which the KCS was very proud of, was the formulation of a domain-based classification scheme for non-life insurance data (See Figure 4-1). With this classification, non-life insurance knowledge can be categorized according to theme and subject. Under these two categories, knowledge items are further categorized in accordance to 12 different topics, all of which are non-life insurance context-specific. The main idea behind the classification is to serve as a cognitive guide for facilitating search for information as well as in describing knowledge in terms of topics that were meaningful for experts. The requirements that were formulated during these workshops will be listed in the next sections.

![Figure 4-1. The classification of non-life insurance knowledge items](image)

With all the ideas for improvement drawn up, and with more explicit requirements articulated, the software engineer from the IT department proceeded with development of the next version of KENNISNET. The version that was put into use has more or less the following architectural implementation (Figure 4-2).

![Figure 4-2. KENNISNET Architecture (Functions and Data)](image)

**4.3.1 Requirements Statements**

Table 4-4 summarizes the requirements derived during this period. Following the account of events in this phase and in recovering the requirements, it is observed that enforcing a process and including the end-users in the design leads to the formulation of more precise requirements.
We consider these requirements to be precise in the sense that they are more specific than the previous requirements. This enhancement in requirements, i.e. generation of more requirements and more precise ones, is shown for example by the software solution specification domain becoming populated with requirements. Each of these statements will be elaborated and discussed.

**EARLY_SS_F_1.** “The initial version of KENNISNET must be improved; it should have more functions and features.” This requirement originates from the outcome of the delivery of the initial version of KENNISNET. In the previous section, we have reported that the initial version of KENNISNET had limited functionalities and was put into trial use only. In the view of the proponents, the system is not satisfactory; therefore, it has to be improved, but they had no idea how. In this instance, there is a clear indication of a requirement because some desire has been made manifest, i.e. a better version of the system is wanted. However, it is rather difficult to formulate it as a specification. This is because while the desired state is known, the means towards achieving it is not well-defined. For example, the specific areas of system improvement were not identified and which additional functions and features the system must have remain unknown. Additional information and details are needed in order come up with a clear write-up, which is a desirable quality of a good requirements specification.

Therefore, while at first glance, the formulation of **EARLY_SS_F_1** (Table 4-4, Item 1) appears to be rather general and vague, it nonetheless gives an indication of an expressed need, but which is not well articulated. Accordingly, we have also classified this requirement to be in the software solution specification requirements domain. This is because the requirement refers to the desirable property of the software as a whole, even though the functional and quality specifications are not well-defined. More importantly, the software application that is of interest here, and which the requirement **EARLY_SS_F_1** is about, is already a finished product. As finished products are defined by their products specifications, **EARLY_SS_F_1** is about the product specifications of KENNISNET. In our conceptual framework, the product specification (product specs) should embody the software solution specification, when completed or acquired. Therefore on the basis these arguments, it is more logical to classify the requirement in the software solution specification domain.

**EARLY_BP_1.** “KCS wants to have a system that the non-life insurance experts will use.” Likewise, this requirement originates from the outcome of the delivery of the initial version of KENNISNET to KCS. The difference, however, between this requirement and the previous one, is the focus and emphasis. **EARLY_BP_1** is oriented more towards describing a desired situation in the environment in which the software is part of. On the other hand, **EARLY_SS_F_1** purely pertains to the software devoid of the context. In these circumstances, this requirement evidently belongs to the business problem domain.

**EARLY_BS_1.** “The end-users should be involved in the re-design of the system in order to come up with more definite requirements.” The choice of a particular design principle, i.e. participatory approach, is requirements specification in the business solution. This is because it prescribes the enactment of a behavioral process which remains to be non-software in nature, even though it is intended to have an impact on the software product.

**EARLY_SS_F_2.** “The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.” This is the specification that translates the knowledge classification formulated during the design workshop (See Figure 4-1). The knowledge items stored in the KENNISNET should reflect the classification. Among the other software specification solution statements in this period, this is a requirement that is more or less reflected in the finished product (See Figure 4-2).

The software solution specification statements from **EARLY_SS_F_2** up to
EARLY_SS_Q_12 (Table 4-4: Items 4 through 13) are requirements we were able to recover on the basis of the written reports that were produced during the design workshops and a demonstration of the resulting KENNISNET system. Most of the specifications are quoted verbatim from the reports and project plans. As indicated in the list, it can be seen that the design team were able to come up with functional, quality or non-functional requirements, as well as operational requirements and design constraints. While some of these specifications are amenable to design implementation, some are not. For example, while EARLY_SS_Q_10 is an ambiguous statement of a requirement, it cannot however be guaranteed by the designer of the system. On the other hand, some of the statements correspond to actual product specifications (Table 4-4, Items 4, 6 through 10, and 14). Figure 4-2 is a supporting representation of these specifications.

Table 4-4. Case A. Requirements at Early Implementation

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_SS_F_1</td>
<td>The initial version of KENNISNET must be improved; it should have more functions and features</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_BP_1</td>
<td>KCS wants to have a system that the non-life insurance experts will use.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SS_F_2</td>
<td>The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop. See Figures 4-1 and 4-2.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SS_F_3</td>
<td>Users should be informed of updates and changes.</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_F_4</td>
<td>The system should enable users to communicate directly with others, either one to one, or many to many.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_SS_F_5</td>
<td>KENNISNET should allows for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory (see Figure 4-2)</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_SS_F_6</td>
<td>It allows users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_SS_F_7</td>
<td>Users are able to know who have posted information, at what time and under which topic and theme.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_SS_F_8</td>
<td>The system should support two knowledge repositories: the knowledge bank and the experts directory. See Figure 4-2.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_Q_9</td>
<td>There should be an online manual.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_SS_Q_10</td>
<td>The user should be able to trust the sources received.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_SS_Q_11</td>
<td>The system should be integrated with other knowledge systems such as ROLLS and DIAGNOSE.</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_SS_Q_12</td>
<td>The system should run on Lotus Notes.</td>
</tr>
</tbody>
</table>

4.3.2 Remarks

While not explicitly shown in Table 4-4 in the list of updated requirements, it must be mentioned here that most of the preliminary requirements remain unchanged. For example, the business problem did not change with the exception of EARLY_BP_1, which indicates that the problem perspective of the KCS has changed. However, the original problem context, the organizational goal, as well as the problem perspective of the other stakeholders – top management and non-life insurance experts, remain unchanged. Likewise, the requirement from the period earlier, such as PRE_SP_1: “The system should be a knowledge management system for enabling knowledge exchange” remain the same and valid for this period (Table 4-3). As also just discussed, some of the statements are actually implemented, i.e. EARLY_SS_O_12: “The system should run on Lotus Notes.” and therefore should be regarded as product specifications. However, the specifications are written as requirements statements which reflect what 'should' be achieved or what is needed to be done.
4.4 Requirements at post-deployment: removal or re-orientation to another use?

The revised version of KENNISNET was officially put into use a few months after the workshop. The system was given to the experts with the idea that they will use it voluntarily. In the beginning, there was an interest in the system and especially those who took part in the design took pro-active efforts to use the system:

I remember at the beginning, I have put information in KENNISNET and started to wait for reaction from my colleagues. – Frans, Product Manager

At the beginning, I looked at KENNISNET. I think once a month, not often. I get some ideas about competitors. It is interesting to know what other colleagues think about it. – Jerome, Product Manager.

However, as the months passed by, KENNISNET usage remained very low which worried KCS. System logs of user activities on the system describe the level of system usage:

• Only 67 new entries were added after a period of 7 months; this translates to approximately 10 new entries per month;
• 31% of all the contributions to the Knowledge Bank were all from Tom who is the project leader of KENNISNET;
• Less than half (46%) of the user base (N=39) have made an entry to the system
• There are over 800 user activity logs related to reading and searching for information within KENNISNET.

Several efforts were made to improve KENNISNET usage such as including it in the agenda in the regular business units meeting as well as in the workshops. The experts were asked if they would have any suggestions on how to improve KENNISNET such that it would be useful to them. Likewise, from time to time, the project leader would send an email to some of them, urging them to make inputs to KENNISNET. One product manager recalls,

Tom asked me once to put information about the product pallets because I made a presentation in the workshop. But you have to realize that it is a lot of work. But he did send me an email after, thanking me for doing it. – Josie, Product Manager

Despite not using the system, the experts still consider KENNISNET to be a useful tool and felt that they needed it:

I think it is a good system and we need it. It can be used for exchanging information. – Paul, Actuary.

...still, I find the system useful. You could ask questions to your colleagues and give answers as well. – Phil, Product Manager.

I find the system supportive and fills in the gaps with respect to my work. It also provides knowledge about your work. It saves you time in searching for information about a colleague in the directory. – Bill, Insurance Technical Specialist/Actuary.

Others, on the other hand, expressed their confusion:

Honestly speaking, I don’t know if I need it or not. I feel my tasks are oriented towards human aspects more than technological ones. I have to talk with other people, negotiate, etc. I believe it is an illusion that the system may perform such functions. KENNISNET cannot do your job because it is too static. – Jerome, Product Manager.
Other means to stimulate system were also introduced. For example, the KCS conducts team-building workshops three times in a year. In one of these workshops, a special session was organised for discussing KENNISNET, i.e. how to improve its use, what kind of information can be stored there, when and who shall input information, amongst others. However, there was very little interest and participation in the special session for KENNISNET during the workshop.

**Purposes and Frequency of System Use**

While KENNISNET is intended for use by the entire group of non-life insurance experts, only a few have indicated that they made use of the system. In addition, their use of the system is limited to occasional use, i.e. they have used the system only once or once in a while when the need arises. This need is indicated by following purposes for which users have indicated they make use of the system:

- To search for information relating to their job, i.e. a news article, a way about calculating premiums, a marketing report, etc.
- To search for a colleague who is an expert on a certain non-life insurance domain, i.e. car insurance, travel insurance.

In both types of use purposes, the essence of use indicates a search-oriented task: the user searches for information, or using their local terms, knowledge. As the figures on user logs indicate, the users carried out only one side of knowledge exchange; that is, acquiring knowledge but not sharing what they know.

**Why Users Don’t Use the System**

In most occasions, the system is not actually used. A number of users have also indicated that they never used or have hardly used the system at all. In the latter case, they have probably used KENNISNET once – just to orient themselves about what the system is about, but did not make any further use of the system. As previously mentioned, some users did make use of the system but due to several reasons had stopped using the system. The reasons given are as follows:

- No replies are given on questions posted
- Other sources of information and means of communication with co-experts are available.
- Quality of information is subject to doubt.
- Information in the system is useful more for product managers than actuaries
- The system is not easy to use: it takes a number of steps before one can classify the information that one wants to enter (using the taxonomy of knowledge items that users have developed or to find the information that one is looking for.

4.4.1 Requirements Statements

In the light of the context characterizing the post-deployment period, we have come up with the following set of requirements statement that reflect the situation (Table 4-5). They also include the suggestions for improvement, some of which were proposed by the users while some were elicited while the case study was being conducted. The suggested improvements cover a wide range of options for enhancements – from system functionalities to soft measures such as group agreements to perform certain actions or enforcing policies.

**POST_BP_1.** “The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.” This statement reflects the changing problem space of KCS in the course of system implementation – from first desiring to come up with a knowledge management system, to having a better system and to having a situation that meets their intentions. This requirement is classified within the business problem domain representing a stakeholder’s
perspective. This requirement is brought about by a breakdown in the software solution specification domain: the system, in its current version and with its own set of functions and features, does not fulfil users’ expectations.

Table 4.5. Case A. Requirements Statements at Post Deployment

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>2</td>
<td>POST_BS_1</td>
<td>The users should be encouraged to make use of the system by organising meetings to talk about the system and finding ways to use it.</td>
</tr>
<tr>
<td>3</td>
<td>POST_BS_2</td>
<td>Users should put training and seminar materials in KENNISNET. If not available in electronic format, a brief summary of the training/seminars/courses should be written and be made available for other colleagues through KENNISNET.</td>
</tr>
<tr>
<td>4</td>
<td>POST_BS_3</td>
<td>Users should publish in KENNISNET market research and competitor reports that were conducted in their sub-companies</td>
</tr>
<tr>
<td>5</td>
<td>POST_BS_4</td>
<td>Users should also publish work methods or processes of how a work problem or project was approached and solved</td>
</tr>
<tr>
<td>6</td>
<td>POST_SP_1</td>
<td>KENNISNET should be converted into a project management system.</td>
</tr>
<tr>
<td>7</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
</tr>
<tr>
<td>8</td>
<td>POST_SS_F_1</td>
<td>The system should provide notification for new entries added to the system.</td>
</tr>
<tr>
<td>9</td>
<td>POST_SS_F_2</td>
<td>The system should allow storage of WWW bookmarks.</td>
</tr>
<tr>
<td>10</td>
<td>POST_SS_F_3</td>
<td>The system should display or make prominent the latest information that is added on the system.</td>
</tr>
<tr>
<td>11</td>
<td>POST_SS_F_4</td>
<td>The system should provide links to information sources such as the Statistical Board.</td>
</tr>
<tr>
<td>12</td>
<td>POST_SS_Q_5</td>
<td>The system should be kept up-to-date.</td>
</tr>
<tr>
<td>13</td>
<td>POST_SS_Q_6</td>
<td>The interface design should be improved; it should be more pleasing aesthetically.</td>
</tr>
<tr>
<td>14</td>
<td>POST_SS_Q_7</td>
<td>Use a more advanced and intelligent search tool like Discovery where search and searcher profiles can be stored.</td>
</tr>
<tr>
<td>15</td>
<td>POST_SS_Q_8</td>
<td>The system should contain information obtained from the Association of Insurance Companies</td>
</tr>
<tr>
<td>16</td>
<td>POST_SS_Q_9</td>
<td>The system should contain product information from each sub-company (brands, tariffs, premiums, claims, clients, internal figures) including planned and proposed.</td>
</tr>
<tr>
<td>17</td>
<td>POST_SS_Q_10</td>
<td>Put the electronic handbooks or handbooks in the system.</td>
</tr>
</tbody>
</table>

The business solution statements POST_BS_1 to POST_BS_3 (Table 4-5: Items 2 through 5) that represents a soft approach towards inducing behavioural adaptation to system implementation. Behavioral changes, such as new ways of working or new task routines, are examples of expected outcomes of a system implementation process, if this happens to be a favourable and useful system. Ways to induce behavioural changes in this case are introducing shared agreements and policies, which are more social and persuasive in nature, instead of letting the system autonomously provoke users to make use of it. Specifically, POST_BS_2 through POST_BS_4 (Table 4-5: Items 3 through 5) are examples of specific formulations of a soft approach, which in turn operationalize the business solution domain of requirements. These requirements were put forward by users as suggestions for making the use of KENNISNET more intensive. They indicate explicit actions that have to be taken by the users. What is important to note about these requirements is how they indicate one particular desirable property users ascribe to the system – content. These requirements indicate what kind of data or information the system must have.

In turn, POST_SP_1 and POST_SP_2 (Table 4-5: Items 6 and 7) are statements that both indicate new software product concepts being assigned to the system. The first requirement, POST_SP_1, is a change in the software product idea without leading to any changes in the
software specifications of KENNISNET. When the concept of a knowledge sharing software system did not work, there were suggestions, especially coming from one member of the KCS to repackage KENNISNET. The idea is to shift the software product idea from a knowledge management application to a shared project management tool, where the cooperative functions intended by the system remain intact. The second requirement, POST_SP_2, calls for a completely different kind of system, in which the functions, content and software idea are totally different. At this phase of the implementation, it became clear that users have a completely different notion of a knowledge management system. The interviews revealed that what users had in mind for a knowledge management is a knowledge resource, i.e. an expert system. For this reason, several of them had suggested the acquisition of COTS-based products such as ROLLS and DIAGNOSE. This software product idea calls for a completely different and new type of system.

The software solution specification statements listed in Table 4-5 as Items 8 through 17 are requirements derived from the suggestions for improvement users have proposed, when asked if they have ideas about how the system can be improved and its use. For example, POST_SS_F_1 to POST_SS_F_4 (Table 4-5, Items 8 through 11) are requirements pertaining to desired functionalities that KENNISNET must have, as expressed by the users. POST_SS_Q_5 up to POST_SS_Q_7 (Table 4-5: Items 12 through 14), on the other hand are quality requirements. The rest of the requirements are data requirements which pertain to desired content, i.e. what kind of information the system must contain. These are statement POST_SS_O_8 up to POST_SS_O_10 (Table 4-5: Items 15 through 17).

4.4.2 Remarks

Out of this compilation from the ex-post reconstruction of events, there is the general observation that requirements gathered span all the requirements domains. Additionally, requirements in the software solution specification domain dominate the space of evolved requirements. This is followed by requirements in the business solution domain, which imply that soft measures were being adopted by the implementing organization.

During the later part of the implementation process, when users begin to interact with the system and try to put it into productive use, the specific and meaningful requirements become apparent. These are indicated by the improved articulation of requirements by users leading to more informative specification of requirements in terms of desired functions and quality. More importantly, at this stage of the implementation the most critical and highly desired requirements become known. In this application domain, viz., knowledge management, some of the most critical requirements are data requirements. But although they are crucial, these requirements are not known or cannot be known in the beginning nor in the early phase of the implementation process.

Apart from new or evolving requirements, what is also observe is the recurrence of unmet requirements. Several requirements that users had articulated during the design and development of the system, i.e. during the second design iteration of KENNISNET, were not implemented in the working version of the system. While these requirement are not really new and that is why they are not on the list, it is important to mention that they were again reiterated by users. For example, the statement EARLY_SS_Q_7 (Table 4-4: Item 9) is an example of a requirement users have mentioned but was not implemented in the revised version of the system that they were using and therefore. They kept looking for that functionality. The same goes for the statement EARLY_SS_F_3 (Table 4-4: Item 5) which states the need expressed by users to be informed of the changes and updates in the system on a regular basis.
4.5 Requirements evolution as within-domain change

Our conceptual framework model requirements evolution descriptively: it is the change in requirements over time. Where these requirements are defined in terms of the different requirements domains (Section 2.7.1; Chapter 3: Figure 3-1), evolution is therefore the change in any of these domains from one time period to another (See Chapter 3: Figure 3-2). Sections 4.5.1 through 4.5.4, compile and analyze the requirements statements per domain from all periods.

4.5.1 Business problem domain evolution

The introduction and eventual use of the KENNISNET system resulted into new business problems. This change can essentially be characterised as shifting problem definitions. The shifts are illustrated in Table 4-6.

Even though the organizational goal did not change over time as well as the goals of the end-users, the problem focus and the stakeholders experiencing a problem through the implementation of the system changed over time. The problem resulting from the implementation of the system is most intensely felt by KCS who is the key system sponsor. The change in their problem definition is centred on the problem focus: from wanting a knowledge sharing system to wanting that knowledge sharing system to be used by its intended users. On the other hand, the goals of the end-users remained the same. They still faced the same kind of problems as before the system was implemented.

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BP_1</td>
<td>Active Insurance Group intends to respond to continuing mergers and acquisitions.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BP_2</td>
<td>One of the goals of the organization is to build a community of employees and to evoke a feeling of ‘one AIG company’</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BP_3</td>
<td>The KCS - non life insurance aims to translate the organizational goal of integrating and unifying the different sub-companies by developing a community of non-life insurance experts</td>
</tr>
<tr>
<td>4</td>
<td>PRE_BP_4</td>
<td>The KCS - non-life insurance is tasked with developing the competency of the non-life insurance experts</td>
</tr>
<tr>
<td>5</td>
<td>PRE_BP_5</td>
<td>The non-life insurance experts would like to perform their tasks efficiently and to be able to gather the information they need from colleagues in other division in an uncomplicated manner</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_BP_1</td>
<td>KCS wants to have a system that the non-life insurance experts will use.</td>
</tr>
<tr>
<td>7</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
</tbody>
</table>

The shift in business problem definition is due to the changing problem perspective a key stakeholder in the process. In this case have instantiated the notion of requirements uncertainty (Harker et al. 1993; Ciborra 1996; Orlikowski 1998), an already known problem is associated with requirements evolution.

4.5.2 Business solution domain evolution

The evolution of requirements statements representing the business solution is compiled in Table 4-7. From pre-implementation up to the post-deployment, a progression can be seen in the way business solutions were sought, conceived and externalised. One striking aspect of this evolution is the way in which solutions tend to move from the level of the organization towards the users and their behavior. Business solutions move away from the formal policies to the more personal and behavioral approaches. This is manifested through the choice for more soft approaches, i.e.
local, informal and decentralized actions. For example, KCS adopted a more consultative approach to system re-design. Organizing meetings with end-users and making agreements about further stimulating the use of the system would all constitute these soft approaches.

Table 4-7. Case A. Business solution statements compilation: all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R )</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements</td>
</tr>
<tr>
<td>3</td>
<td>POST_BS_1</td>
<td>The users should be encouraged to make use of the system by organising meetings to talk about the system and finding ways to use it.</td>
</tr>
<tr>
<td>4</td>
<td>POST_BS_2</td>
<td>Users should put training and seminar materials in KENNISNET. If not available in electronic format, a brief summary of the training/seminars/courses should be written and be made available for other colleagues through KENNISNET.</td>
</tr>
<tr>
<td>5</td>
<td>POST_BS_3</td>
<td>Users should publish in KENNISNET market research and competitor reports that were conducted in their sub-companies</td>
</tr>
<tr>
<td>6</td>
<td>POST_BS_4</td>
<td>Users should also publish work methods or processes of how a work problem or project was approached and solved</td>
</tr>
</tbody>
</table>

4.5.3 Software product concept domain evolution

For this domain, it is observed that using and interacting with the system led to a change in the product idea assigned to it. This is shown by the compilation of requirements that pertain to the software product idea in Table 4-8.

Table 4-8. Case A. Software product concept statements compilation: all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R )</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
</tr>
<tr>
<td>2</td>
<td>POST_SP_1</td>
<td>KENNISNET should be converted into a project management system.</td>
</tr>
<tr>
<td>3</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
</tr>
</tbody>
</table>

We can see from Table 4-8 that the shift in software product concept is most pronounced in the later part of the implementation in this case. This is at the post-introduction phase of the implementation when users had a longer period of engagement with the system. There are two types of shift that can be seen here. The first is the shift in the software product concept without leading to drastic changes in the software solution specification (POST_SP_1; Table 4-8, item 2). The second type of shift is one in which a totally new product is called for and the currently being used system needs to be replaced (POST_SP_2; Table 4-8, item 3). This new software product concept will then have a different set of solution specifications than the ones generated for KENNISNET.

4.5.4 Software solution specification domain evolution

The solution space of the software specification did change in the course of implementing and using KENNISNET at AIG. The evolution of requirements in this domain can be characterised by the following attributes: (i) increased number of specifications, and (ii) improved information content of the specifications. In other words, more explicit specifications emerge as users begin to articulate them. These can be seen in the requirements specifications compiled in Table 4-9.
Table 4.9. Case A. Software solution specification statements: all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SS_F_1</td>
<td>The system should store and remember data</td>
</tr>
<tr>
<td>2</td>
<td>PRE_SS_O_2</td>
<td>The system should be developed in Lotus Notes</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SS_F_1</td>
<td>The initial version of KENNISNET must be improved; it should have more functions and features</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SS_F_2</td>
<td>The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop. See Figures 4-1 and 4-2.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SS_F_3</td>
<td>Users should be informed of updates and changes</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_F_4</td>
<td>The system should enable users to communicate directly with others, either one to one, or many to many.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_SS_F_5</td>
<td>KENNISNET should allows for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory (see Figure 4-2)</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_SS_F_6</td>
<td>It allows users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_SS_F_7</td>
<td>Users are able to know who have posted information, at what time and under which topic and theme.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_SS_F_8</td>
<td>The system should support two knowledge repositories: the knowledge bank and the experts directory. See Figure 4-2.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_O_9</td>
<td>There should be an online manual.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_SS_O_10</td>
<td>The user should be able to trust the sources received.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_SS_O_11</td>
<td>The system should be integrated with other knowledge systems such as ROLLS and DIAGNOSE.</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_SS_O_12</td>
<td>The system should run on Lotus Notes</td>
</tr>
<tr>
<td>15</td>
<td>POST_SS_F_1</td>
<td>The system should provide notification for new entries added to the system.</td>
</tr>
<tr>
<td>16</td>
<td>POST_SS_F_2</td>
<td>The system should allow storage of WWW bookmarks.</td>
</tr>
<tr>
<td>17</td>
<td>POST_SS_F_3</td>
<td>The system should display or make prominent the latest information that is added on the system.</td>
</tr>
<tr>
<td>18</td>
<td>POST_SS_F_4</td>
<td>The system should provide links to information sources such as the Statistical Board.</td>
</tr>
<tr>
<td>19</td>
<td>POST_SS_O_5</td>
<td>The system should be kept up-to-date.</td>
</tr>
<tr>
<td>20</td>
<td>POST_SS_O_6</td>
<td>The interface design should be improved; it should be more pleasing aesthetically.</td>
</tr>
<tr>
<td>21</td>
<td>POST_SS_O_7</td>
<td>Use a more advanced and intelligent search tool like Discovery where search and searcher profiles can be stored.</td>
</tr>
<tr>
<td>22</td>
<td>POST_SS_O_8</td>
<td>The system should contain information obtained from the Association of Insurance Companies</td>
</tr>
<tr>
<td>23</td>
<td>POST_SS_O_9</td>
<td>The system should contain product information from each sub-company (brands, tariffs, premiums, claims, clients, internal figures) including planned and proposed.</td>
</tr>
<tr>
<td>24</td>
<td>POST_SS_O_10</td>
<td>Put the electronic handbooks or handbooks in the system.</td>
</tr>
</tbody>
</table>

4.6 Requirements evolution as inter-domain change: impact relations

Besides substantiating requirements evolution by providing evidences of requirements changes in each of the domains, we have also identified a pattern of requirements change. This pattern is made apparent by a series of impact relations that were repeatedly observed among the requirements in this case. Change is effected through these impact relations and it revolves around the following pattern: requirements in one domain in the requirements matrix have an impact on the requirements in another domain. Following this pattern, the requirements in the affected domain will shift.
An impact relation is dynamic link between two domains whose requirements are related to each other. This relation is instantiated when two or more requirements that belong to different domains are linked together, where one requirement impacts a requirement or more in another domain. It is the term we would like to use to refer to a frequently observed chain of events that cause a change in requirements. This process of change follows a sequential, cross-domain path. It is also usually the case that the source requirement has an underlying issue affecting it and this leads to the formulation of a new requirement as a way of resolving the issue. An issue is hereby understood as a concern or interest stakeholders attach to a requirement with a certain amount of importance which demand that action or decision be taken in order for the issue to be settled or addressed. To further illustrate these observations, we will go through an example from the case. In Figure 4-3, we can see two requirements that reflect a sequence of events. The first requirement in the left quadrant, PRE_BP_1, is a requirement in the business problem domain. The second requirement PRE_BS_1 on the right quadrant is a requirement in the business solution domain. The arrow shows the impact relation between the two.

On the basis of this case alone, we have identified seven types of change sequences in how requirements impact each other. These types are shown in Figure 4-4 and are further discussed in the succeeding paragraphs with supporting evidence from the case. We will denote each type by the notations M1 to M7. Table 4-10 provides the textual description of the 7 impact relations.
In the succeeding sections, we will discuss the different impact relations observed in this case. A compilation of the different impact relations can be referred to in Appendix C to I.

4.6.1 M1. Business problem resolved by a business solution

We have encountered this mechanism all throughout the different periods in the case – from pre-implementation through post-deployment

- **PRE_BP_2.** “One of the goals of the organization is to build a community of employees and to evoke a feeling of ‘one AIG company’” resolved by PRE_BS_1: “The adoption of a Knowledge Management (KM) strategy is a means of community-building”; PRE_BP_3: “The KCS - non life insurance aims to translate the organizational goal of integrating and unifying the different sub-companies by developing a community of non-life insurance experts” resolved by PRE_BS_1. (See Appendix C: Case A, Items 1-2)

In Figure 4-3, we have already cited and elaborated on the resolution of the requirement PRE_BP_2 by PRE_BS_1 as an instance of an impact relationship. At the same time, PRE_BS_1 is also a resolution to another requirement representing the problem definition of a
key stakeholder, namely top management. This is the statement requirement PRE_BP_3.

- EARLY_BP_1. “KCS wants to have a system that the non-life insurance experts will use.” resolved by EARLY_BS_1. “The end-users should be involved in the re-design process in order to come up with more definite requirements.” (See Appendix C: Case A, Item 3)

At early implementation, a business problem by the KCS, captured by the statement EARLY_BP_1 was resolved through EARLY_BS_1, which is the decision to implement participatory design process (See Section 4.6).

- EARLY_BP_1. “KCS wants to have a system that the non-life insurance experts will use.” resolved by POST_BS_1. “The users should be encouraged to make use of the system by organising meetings to talk about the system and finding ways to use it.”

In the same principle, POST_BS_1 is an attempt at resolving the persistent need of the system proponents to have KENNISNET used by the product managers and actuaries.

- POST_BP_1. “The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts” resolved by statements -- POST_BS_1: “The users should be encouraged to make use of the system by organising meetings to talk about the system and finding ways to use it.”; POST_BS_2. “Users should put training and seminar materials in KENNISNET. If not available in electronic format, a brief summary of the training/seminars/courses should be written and be made available for other colleagues through KENNISNET.”; POST_BS_3. “Users should publish in KENNISNET market research and competitor reports that were conducted in their sub-companies.”; POST_BS_4: “Users should also publish work methods or processes of how a work problem or project was approached and solved.” (See Appendix C: Case A, items 4-7)

In the latter part of KENNISNET implementation, the impact relation M1 is manifested in various instances. Specifically, it is driven by one key business problem, POST_BP_1 being resolved by various business solutions. For example the impact relation POST_BP_1 and POST_BS_1 is motivated by the desire of KCS to improve the state of use of the system. They want KENNISNET to be intensively used. In order to bring about this desired outcome, they organised meetings and talks with the end-users in the hope that these will encourage them to use the system. Similarly, this problem statement is also resolved by yet another requirement in the form of POST_BS_2. Aside from encouraging the end-users to make intensive use of the system, specific form of actions were also put forward, i.e. users should make training and seminar materials available for access in the system. Next to this, two more specific actions resolutions were thought of in order to further solicit end-user cooperation in using the system. These are given as statement POST_BS_3 and POST_BS_4, respectively. Both of these require end-users to externalise their knowledge through the system in definite terms by (i) publishing market reports they have written and (ii) making known the method or process they have developed or applied when handling a new project or solving a certain problem.

4.6.2 M2. Business solution motivating a software product concept

This impact relation is the immediate driving force that led to the development of the system and in later periods, it provides ideas for alternative software products. It is given by the following set of requirements.

- PRE_BS_1. “The adoption of a Knowledge Management (KM) strategy is a means of community-building.” realized by PRE_SP_1: “The system should be a knowledge management system for enabling knowledge exchange.” (See Appendix D: Case A, item 1)
As mentioned in the account of events surrounding the implementation, KCS took the initiative of further realising the knowledge management strategy of the organization (PRE_BS_1) by implementing a software solution in general (PRE_SP_1).

- EARLY_BS_1. “The end-users should be involved in the re-design process in order to come up with more definite requirements.” realized by POST_SP_1: “KENNISNET should be converted into a project management system.” and POST_SP_2. “KENNISNET should be in the form of an expert system.” (See Appendix D: Case A, Items 2 and 3)

In later phases, the decision to include end-users in the design process give rise to ideas for other types of knowledge applications, which the users themselves have suggested.

4.6.3 M3. Software product concept realized by a chosen software solution specification

This relation corresponds to the rational process of enacting a design decision to a chosen product idea. It must be kept in mind that the enacted design decision consisting of solution specifications is just one of the many possible implementations of the product concept. In this case, the following requirements pairs manifest this impact relation.

- PRE_SP_1. “The system should be a knowledge management system for enabling knowledge exchange.” enacted by PRE_SS_F_1. “The system should store and remember data.”; PRE_SS_O_2. “The system should be developed in Lotus Notes.”; EARLY_SS_F_2. “The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.”; EARLY_SS_F_3. “Users should be informed of updates and changes.”; EARLY_SS_F_4. “The system should enable users to communicate directly with others, either one to one, or many to many.”; EARLY_SS_F_5. “KENNISNET should allow for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory.”; EARLY_SS_F_6. “It should allow users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.”; EARLY_SS_F_7. “Users are able to know who have posted information, at what time and under which topic and theme.”; EARLY_SS_F_8. “The system should support two knowledge repositories: the knowledge bank and the experts directory.”; EARLY_SS_Q_9. “There should be an online manual.”; EARLY_SS_Q_10. “The user should be able to trust the sources received.”; EARLY_SS_Q_11. “The system should be integrated with other knowledge systems such as ROLLS and DIAGNOSE.”; and EARLY_SS_Q_12. “The system should run on Lotus Notes.” (See Appendix E: Case A, Items 1 through 13)

The software product concept given by PRE_SP_1 calling for a system that will facilitate knowledge exchange is implemented in various ways from pre-implementation to early implementation. At pre-implementation, the knowledge sharing system that KCS had in mind, was worked out in terms of two loose design specifications: (i) the system must have a database and (ii) it must be built using a certain development environment such as Lotus Notes. The same product idea continued to prevail in the early implementation period which is further realized by the following statements EARLY_SS_F_2 to EARLY_SS_Q_12 (Table 4-4: Items 4 through 14) that were formulated from the design workshops.

- POST_SP_2. “KENNISNET should be in the form of an expert system.” conveyed by POST_SS_Q_7. “Use a more advanced and intelligent search tool like Discovery where search and searcher profiles can be stored.”; POST_SS_O_8. “The system should contain information obtained from the Association of Insurance Companies.”; POST_SS_O_9. “The system should contain product information from each sub-company (brands, tariffs, premiums, claims,
4.6 Requirements evolution as inter-domain change: impact relations

clients, internal figures) including planned and proposed.” (See Appendix E: Case A, Items 14 through 16)

We have seen in the account of the events that took place in the case that the software product concept had evolved. From a knowledge management system, there were proposals to change the system into an expert system (POST_SP_2), with suggestions to realize this idea ranging from an intelligent search tool into an information database, i.e. POST_SS_Q_7, POST_SS_O_8.

4.6.4 M4. Software solution specification supported by a business solution

This is an occurrence where the implementation of a specific design solution triggers an action in the business solution domain. This action could be in the form of initiating and carrying out a new process, i.e. new tasks or a development approach, and formulating and implementing a new policy about system design and use, as well as making informal agreements about these. In other words, the actions triggered in the business solution domain call entail some form of behavioral adaptations in the business and social environment.

We found the following requirements pair that manifests this impact relation. It has also been observed that this condition is evident only after the system has been deployed and used in the organization.

• EARLY_SS_F_1. “The initial version of KENNISNET must be improved; it should have more functions and features” supported by EARLY_BS_1: “The end-users should be involved in the re-design process in order to come up with more definite requirements.”(See Appendix F: Case A, Item 1)

The requirement EARLY_SS_F_1 was articulated when the initial version of the system was delivered. It specifies that the initial version of the system must be improved by containing more functions and features. This is resolved by KCS through the adoption of a participatory design process which entails the involvement of the end-users, namely the non-life insurance experts. This action is captured by the requirement EARLY_BS_1.

4.6.5 M5. Business solutions leading to improved software solution specifications

This impact relation describes how actions taken in the business solution domain can lead to improvements in the formulation of software solution specifications. This improvement in specifications, i.e. becoming more informative and enriched, has been substantiated in the previous section as one dimension of requirements evolution. In the mean time, this relation is given by the following requirements whose conception is related to the previous impact relation, M4. The connection between the following requirements and the requirements in the relation M4 will be made clear in the ensuing discussion of the impacts.

• EARLY_BS_1. “The end-users should be involved in the re-design process in order to come up with more definite requirements.” leads to the formulation of EARLY_SS_F_2. “The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.”; EARLY_SS_F_3. “Users should be informed of updates and changes.”; EARLY_SS_F_4. “The system should enable users to communicate directly with others, either one to one, or many to many”; EARLY_SS_F_5. “KENNISNET should allows for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory.”; EARLY_SS_F_6. “It allows users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.”; EARLY_SS_F_7. “Users are able
to know who have posted information, at what time and under which topic and theme.”; EARLY_SS_F_8. “The system should support two knowledge repositories: the knowledge bank and the experts directory.”; EARLY_SS_Q_9. “There should be an online manual”; EARLY_SS_Q_10. “The user should be able to trust the sources received.”; EARLY_SS_Q_11. “The system should be integrated with other knowledge systems such as ROLLS and DIAGNOSE.”; and EARLY_SS_O_12. “The system should run on Lotus Notes.” (See Appendix G: Case A, Items 1 through 11)

As mentioned in Section 4.3 the participation of the end-users in the re-design of the system led to an improvement in the specification of requirements for the knowledge management system. The improvement is in terms of increased number of requirements and increased information content in the specification in the sense that these become less vague. In other words, the specifications become more prescriptive which makes them a step amenable to design efforts.

This way, we can say that the chosen business solution EARLY_BS_1 in the form of a decision to make use of a participatory design approach has brought about improvements in the solution specification domain. Six new requirements were formulated out of this process. Half of these requirements are instructive functional specifications and the other half are explicit statement of desired qualities (Table 4-4). Furthermore, these are impact relations that are closely associated with each other. Had it not been for the process that was implemented (EARLY_BS_1), the new requirements in the software solution specification domain at early implementation would have not been formulated.

• PRE_BS_1. “The adoption of a Knowledge Management (KM) strategy is a means of community-building.” resolved by EARLY_SS_F_2. “The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.” and EARLY_SS_F_4. “The system should enable users to communicate directly with others, either one to one, or many to many.”

While formulated in the beginning of the implementation, PRE_BS_1 is also fulfilled by specifications formulated at a later period.

4.6.6 M6. Software solution specification leading to a new business problem

This relation is a reverse sequence of events with respect to the conventional view that problems precede solution. The following combination of requirements indicates that solutions can lead to problems.

• Product specifications implemented from statements – EARLY_SS_F_2 to EARLY_SS_O_12 (Table 4-4: Items 4 through 14) leading to POST_BP_1. “The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.” (See Appendix H: Case A, Items 1 through 11)

In this impact relation, the resulting software product based on the specifications formulated from the design workshop leads to a new business problem. This business problem is given by the requirement POST_BP_1. This problem is a problem statement of only one stakeholder, namely KCS. The low level of use of KENNISNET was an outcome that KCS was not expecting considering the enthusiasm shown by the end-users in participating at the design workshops. Therefore, KCS would like to change this situation. This problem is significant because it has a high level of importance for KCS, especially for one of its members, the project leader for KENNISNET; he was hired specifically for the KENNISNET project.
4.6.7 M7. Business problem resolved by redefining a software product concept

This is an impact relation that shows the resolution of a problem or a newly created problem due to a solution, by changing the assigned product concept to an already existing software product that is already in use. This relation is made conspicuous by the following requirements.

- POST_BP_1. “The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.” resolved by POST_SP_1. “KENNISNET should be converted into a project management system.” and POST_SP_2. “KENNISNET should be in the form of an expert system.” (See Appendix I: Case A: Items 1 and 2)

Related to the previous impact relation example (M6), the resulting new business problem POST_BP_1 is further resolved in this case through formulation of two new product ideas. The first product idea (POST_SP_1) calls for a shift from knowledge management system to a project management system for the implemented and used version of KENNISNET. For this new product idea, it need not necessarily entail changes to the solution specification of the chosen solution concept. In other words, the software need not dramatically change in terms of its functions, components and quality aspects. Another resolution to the business problem by way of attributing a new product idea is through POST_SP_2. This requirement calls for a shift in product idea from a knowledge management to an expert system. In contrast to the former product idea, this product idea calls for a completely new system in which a different set of software solution specification are needed. Should this software product idea get to be implemented, it will lead to the imminent replacement and removal of KENNISNET.

On the whole, these seven configurations of impact relations that we have come upon in this study reflect the dynamic, process-based aspects of requirements evolution. They indicate how requirements evolve by tracing the transition from one domain to another through the requirements pairs. On the basis of this analysis too, we have also seen that the impact relationship need not be an exclusive one-to-one correspondence with one requirement from one to domain to the other. One requirement can impact many requirements in another domain.

4.7 Discussion

There are several points that need to be discussed as far as results of this analysis are concerned and in terms of the analytic explanations they offer. To do this, we would like to talk about the impact relations (Section 4.7.1), review the case study propositions and update our model (Section 4.7.2) and offer some lessons learned (Section 4.7.3)

4.7.1 Impact relations: dynamics of requirements evolution

The impact relations were derived from the requirements that were explicitly distinguishable to be impacting each other. Therefore, not all of the requirements that we have derived are impacting each other. Some requirements in one domain do not lead to an update in terms of a new requirement in another domain. In addition, the requirements that we have derived are only those which we can make explicit on the basis of case study material. It could be the case that there are other requirements that we are not able to identify. Our requirements list may therefore not be exhaustive.

Second, the impact relation usually has a sequential alignment in the sense that one requirement follows the other. The former is the source of the impact and the other is the update. However, the impact relation is not an explicit causal relation. It is more like the source requirement is earlier articulated than the update. The update is a response to the issue raised by the source requirement. To identify an impact relationship, we first take a requirement in one
domain and see whether it has influenced a requirement in another domain. This establishes the impact relation and not vice-versa.

Third, and this is related to the previous point, the problem definitions and solution specifications that we have come up with are only those that are related to the system and its implementation. Of course the organization has other problems and had probably come up with solutions that are not related to the implementation of the system. However, we covered only the concerns that are important for the stakeholders in relation to the system that was implemented.

Fourth, while these impact relations describe the dynamic aspects of requirements evolution, they do not indicate what triggered the requirements to change. At most, they indicate the path of change or the link between two domains where a requirement from one domain is the source of a requirement in the other. What we have noticed and this we have already acknowledged earlier is the presence of an issue surrounding a requirement. The affected requirement prompts a resolution in another domain through a new requirement. Not all requirements however have issues attached to them. These issues surrounding a requirement which triggers or instantiates the impact relation will be an added focus in the next case study.

Fifth, and this is to conclude, with this analysis, we can redefine evolution in terms of process descriptions. We backtrack a bit from the initial definition of evolution given in 2.7.2 where we define evolution as a state of change. It is a definition of evolution as a noun, where it indicates a state of change in requirements in any of the domains. If the requirements in any of the domains have changed, we already say that evolution has already taken place. We have given our observations of this in Section 4.5 where we have characterized the change in each of the requirements domains in the requirements matrix.

On the other hand, with additional knowledge about impact relations, we can define evolution also as a process of change. The notion of evolution is qualified by adding the dynamic aspect of the process in the definition. Utilizing the pattern of change we have earlier mentioned, we can say that evolution is the result of an impact relation between requirements in different domains. Therefore, we make a distinction between two types of changes: a within-domain change and inter-domain change. In this case, evolution has taken place when there are changes in the requirements domains. Or in a more restrictive perspective, we can say that requirements evolution has taken place when a requirement in one domain has led to a new or changed requirement in another domain.

4.7.2 Review of case study goal, questions and conceptual framework update

The application of the conceptual framework in analyzing the case was useful in learning about the dynamics of requirements change, which is the goal we set for this study. Together with the guidance of the level 2 questions for individual cases (Section 3.3.4, p. 48) in gathering data, we come up with the following conclusions based on this pilot study:

- Requirements can be identified and classified according to different domains (See Sections 4.2 through 4.4).

We were able to arrive at this conclusion by gathering data that addresses case study questions L2Q1 up to L2Q3 (Section 3.3.4, p. 48). These questions basically ask what the requirements are in terms of the four requirements domains and the stakeholders involved.

- Introducing a new groupware application leads to requirements evolution in the form of change to any of the requirements domain.

Through the data provided by L2Q4: “What has changed in the requirements ever since the groupware was implemented and how do the changes affect the requirements domains?”, we
were able to observe and focus on changes to requirements at different phases in implementation. This is carried out in the analysis on sections 4.2 to 4.5.

- Evolution can be a within-domain change or a inter-domain change. Inter-domain changes are brought about by impact relations between two requirements belonging to different domains.

This conclusion summarizes the data gathered from addressing the question of what emerging patterns arise from the observed changes in requirements (L2Q5; 3.3.1 p. 46). Answering this question introduced us the concept of impact relations, which brought our attention a specific kind of change taking place as interactions between requirements domains.

- An impact relation can be triggered by an issue affecting a requirement in one domain, prompting an update in another domain. An issue is defined as an important question attached to a certain requirement necessitating decision or action. It is therefore a stimulus for change.

Specifically in the discussion of various instances of impact relations and in trying to identify factors that contribute to change (Section 3.3.4, p. 48, L2Q5) throughout the case, we have come across the notion of an issue that affects a requirement in one domain, and eventually finds update in another domain. The update materializes because a stakeholder took action or have contemplated into taking action in order to address the issue. With this finding, we can reformulate L2Q5 into a more specific case study question:

L2Q5: What are the issues that affect a requirement that prompts a change or update in another domain?

Conceptual framework: impact relations and updated definition of requirements change

Going back to the conceptual framework (with L2Q7 in mind), this case provide us with the following theoretical and practical implications:

- An update on the conceptual framework by introducing concepts such as impact relations thus providing us with a redefinition of requirements change which is updated from the earlier definition in our initial model (Section 2.7.2 and 3.3.1; Figure 2-6)
- In turn, with the knowledge of impact relations, we can ask more specific case study questions for which our list of level 2 questions can be further updated. We can modify L2Q4 to reflect the different types of change:

  L2Q4: What has changed in the requirements (based on L2Q1) ever since the groupware was implemented and how do the changes affect the requirements domains?

  L2Q4a: What are the changes in each domain across the phases (within-domain change)?

  L2Q4b: What are the impact relations?

- At the practical side, we have also updated the data collection protocol from this case to code requirements statements according to phases and uniquely classify them into domains (see Table 4-2).

More importantly, the contribution of this study with respect to theory development is establishing the usefulness of the conceptual framework in our initial attempt to deal with case data. This usefulness is demonstrated in the generation of data to represent requirements in terms of the four domains. In addition, it also helped in discovering new concepts and in redefining requirements change, which means, we can further make use of this framework in investigating
additional cases.

4.7.3 Lessons Learned

Aside from learning about the usefulness of the conceptual framework as a theoretical tool for understanding requirements evolution, this case study offers additional lessons learned on the practical side of software implementation. We have taken this point into consideration in our our study questions L2Q7 (Section 3.3.4, page 48) in which inquired about lessons that can be learned from the case either as a challenged or worthwhile implementation of groupware.

As a challenged implementation of a groupware application in this case, there several lessons that can be learned which can provide insights that can be useful in practice. For example, these insights can be used to inform a requirements elicitation process for voluntary cooperative systems and to structure a system implementation process for these kinds of systems. We identify these lessons as follows.

Disparate mental models of system use and intent is a stumbling block

One of the biggest drawbacks of the system design that is apparent is the disparity in the mental models of system use between the system proponents and system end-users. The different stakeholders, the KCS and the non-life insurance experts, had different mental models regarding the system. These are represented through different metaphors of the system image that they evoke when they perceive the system (See Table 4-11).

<table>
<thead>
<tr>
<th>User groups</th>
<th>Role</th>
<th>Mental Model</th>
<th>System Metaphor</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCS</td>
<td>System proponent and user</td>
<td>Mental model based on intent</td>
<td>Community-building system; Knowledge sharing platform</td>
</tr>
<tr>
<td>Non life insurance experts</td>
<td>End-user</td>
<td>Mental model based on utilization</td>
<td>Information resource; Search engine</td>
</tr>
</tbody>
</table>

The system proponent, KCS, proceeded with the implementation of the system based on a mental model of an intention for a knowledge-sharing platform. Out of this, they formed the expectation that the system will be used for sharing knowledge, that the experts would make use of it to create and publish information to share with their colleagues.

On the other hand, the end-users, the non-life insurance experts, were operating under the mental model of utilization. Their view of the system is in the form of an information resource metaphor in the vivid form of a search engine or an expert system. This utilization mental model is confirmed by user activity logs saved by system. The logs recorded a ratio of 85% user activities related to reading and searching, and 15% related to writing or creating files. This shows that users preferred only one side of knowledge exchange, mainly to get information. In addition, as utilisers of the system, the experts also never saw themselves as fillers of information of the system.

User involvement is necessary but not a sufficient condition for prolonged use

This is one of the important lessons learned in this case. Other design factors are more crucial in this particular case such as having the necessary knowledge content stored in the system that is needed by the users. User involvement in this case had effectuated a somewhat positive end-user attitude towards the system despite of its limitations but such behaviour however is not enough to stimulate a continued system use.
Technology cannot be used to stimulate knowledge sharing

Given obstacles to knowledge sharing such as organizational culture, technology cannot be used to stimulate knowledge sharing. In this particular case, there are still several organizational issues such as distributedness and a past history of competition that hamper knowledge-sharing processes among the users.

Successful implementation is critically dependent on the cooperation within the end-user group.

Hence the design process should have focused on end-users’ needs and desires. The system was conceived to accommodate the needs of different stakeholders, while in practice it mainly served KCS. This may have been the cause of the second major failure: the context in which the system was to be used has not been properly addressed. The requirements analysis concentrated on the technical properties of the system, the desired functionality. It was overlooked that the affordances of the system, even if the user interface had been perfect, did not fit well into the everyday tasks of the users.

Social and contextual factors such as power and dissonance play an important role in knowledge management efforts

In a separate analysis of the case, Pumareja and Sikkel (2005) have offered the perspective of power dissonance in understanding why knowledge management systems implementations fail. According to their analysis, knowledge sharing can be seen a process of relinquishing power because one’s knowledge is a source of power. This notion is applicable especially in the case of knowledge-based work where employees are rewarded on the basis of what they know. In the case of the non-life insurance experts, their expertise – their knowledge, is the source of their leverage in the organization, and it is what they are paid for.
CHAPTER

5

CASE STUDY B: FIRST CLASS EXPERIENCE @ CENTRAL CITY COLLEGE

From groupware to eduware

Proceeding from a failed implementation of groupware technology as an initial study, this chapter continues with the next case of a different implementation outcome. The case is about an optimal implementation of groupware which has been in continuous use by a distributed team of teacher-trainers at a postgraduate institute of large city college. Specifically, the groupware application is a collaboration platform serving as the nation-wide communication infrastructure for the teacher-trainers employed by the institute.

Our goal in this case is to further apply the updated conceptual framework (Sections 2.7, 3.3.1, and 4.7.2) and to establish its theoretical usefulness using a different empirical setting. This case also makes use of the updates and improvements, i.e. codes and notations introduced in Chapter 4 (Section 4.2.1) in analyzing this case. Specifically, we would like to learn more about impact relations (see Section 4.6) and how this case manifests these change mechanisms. The resulting case study write up is as follows. Section 5.1 gives an overview of the case, its background, implementation setting, stakeholders, methods and protocols. Sections 5.2 to 5.4 are the requirements statements derived from the case according to the temporal scheme. Sections 5.5 and 5.6 are analyses of requirements change as within domain evolution and as impact relations. Section 5.7 is an in-depth discussion of the new insights introduced by this case to the conceptual framework (Section 5.7.1) for which we revisit the previous case to update the analysis (Section 5.7.2) and closes this case with a review of the goals and an overview of the latest version of the conceptual framework (Section 5.7.3).

5.1 Case background

Central City College (C3) is one of the largest institutions that provide tertiary level education in the Netherlands. It offers approximately 70 different study program to at least 10,000 new students each year. The school is one of the only three institutions that provide teacher training in special education at the postgraduate level in the Netherlands. In turn, the school's Faculty of Education attracts a yearly enrolment of at least 3,000 students and is the most popular choice of study among its students.

5.1.1 Implementation setting: Orthopedagogical Institute

The provision of postgraduate certification in special education is carried out within C3 by the
Orthopedagogical Institute (OI). OI is one of the departments under the Faculty of Education of C3. It is responsible for the provision of education and training for teachers who would like to teach special education. Special education is generally understood as providing instruction to students, in this case primary school children, with special needs, i.e. autistic, dyslexic, handicapped – blind, deaf or mute, learning disabled, among others. To carry out this didactic task professionally, a teacher needs to have a postgraduate diploma in special education. Likewise, to be eligible for this study, he or she needed also to have completed a bachelor’s degree in education or are already teaching. Next to the providing schooling in special education, the institute also provides in-house training and consulting services to primary schools nationwide that have a special education component. Its clients include primary education teachers, special education teachers, primary schools and to a certain extent private individuals. The professional background of its teaching personnel is diverse – some of them are psychologists, teachers with master specializations or teachers with extensive teaching experience. Therefore, from time to time, the institute also performs clinico-psychological diagnosis of children with mental and emotional problems. This institute is one of the departments within C3 that functions autonomously as an organization of its own.

As a very large organization, most of the operations within C3 are decentralized. Each faculty is responsible for it own budget as well as information technology (IT) services. It is for this reason that OI has its own IT department. It also operates as an independent organization although some administrative functions are coordinated with the central body, i.e. personnel department. To carry out its educational activities and support its administrative processes, OI utilizes collaboration technology in the form of an e-learning and content management software. Specifically, it uses a commercial off-the-shelf (COTS) software package called First Class Client, also called FCC (Open Text Corporation 2005).

As an institute that operates at the national level, OI has regional and local offices all over the country. This organizational set-up is consistent with manner in which most continuing education programs have come to be organized: learners receive instruction at the place where they live and work. As a result of this organizational set-up, the teaching personnel of the institute are also distributed all over the country where they work locally; their office is their home. These teacher-trainers are examples of teleworkers. The Institute provides them with a home-office facility that includes furniture and fixtures, computer and software. In turn, these teacher-trainers have to provide space in their homes as work area. Included in the work facilities that they receive is FCC, which is intended to be a tool support for carrying out tasks in a distributed manner. The software also serves as the de facto communication and information exchange platform for coordination and collaboration since the institute had stopped reimbursing telephone costs when it began implementing FCC.

When this case study was conducted, FCC had already been in a phase of sustained use at OI for approximately 5 years. Facilitating this is the continued effort of the department in keeping itself up to date with the latest versions of the software as well as with the latest developments in software technologies, i.e. new Windows operating systems. Because of this prolonged long-term use of the software, we consider this case to be a successful implementation of groupware technology. FCC users at IO consist of employees and students. The employees are made up of both teaching and non-teaching personnel. For purposes of focus and manageability of scope, we conducted the case study only among the teaching staff of IO. This is also because it is within this group of users that the aspect of cooperation is strongest.

5.1.2 Stakeholders

The stakeholders in this case can be divided into two groups: the system proponents represented by the system administrators and system users consisting mainly of the special education
teacher-trainers.

System proponents: FCC administrators

FCC administrators belong to the IT and Helpdesk Department within OI. The department is composed of an ICT manager and three system administrators. Of the 3 FCC administrators, one is a senior administrator and is the key contact for this case study. The ICT Manager together with the senior FCC administrator were main the proponents of the system for the institute. They are the ones who started with its implementation.

Altogether, the main task of these administrators is to ensure that FCC is functioning properly for the entire Institute. They are mainly responsible for the technical aspects of maintaining the system. Since FCC is a COTS application, these tasks do not require intensive coding, but rather consists of system administration, helpdesk and support tasks such as:

- Manage user accounts, i.e. setup, create, maintain, manage access rights
- Provide assistance to users who need help when using the system, either by answering the queries per telephone, answering to their emails or visit the place of work of the employee to troubleshoot technical problems
- Organize and give tutorials to new users and other users who wants to learn new functionalities about the system
- Promote the use of the system by finding ways of how to use the system and other ICT application to improve the work processes within the Institute
- Develop customized user manuals
- Give demonstrations of how the system works to other interested users and clients of the institute
- Implement the digitalization policy for all educational programs which include providing assistance in developing instructional materials, transforming study materials into electronic form, etc.

From this list, it can be seen that aside from being responsible for day to day operations, the administrators are also responsible for initiating IT-enabled innovations at C3.

System users: special education teacher trainers

The greater bulk of the stakeholders are composed special education teacher-trainers that make up the teaching staff of the institute. Their task consists mainly two parts – teaching and consulting. The teaching part consists of all the activities surrounding the provision of educational instruction such as lesson planning, curriculum development, content build-up course evaluation and actual lecturing itself. The consulting part is oriented towards assisting primary and secondary schools towards capacity-building and management of their teaching force.

The work of these teacher-trainers takes place in telework context (Limburg 2002; Olson 1989) manner: it is completely virtual, distributed and decentralized. Resulting from this context is flexible work structure that is not bound by an 8 to 5 schedule. Sometimes, an instructor finds herself giving a 3-hour lesson in the morning or in the evening, or attending a meeting with the administrators of the school. When not attending meetings, giving lessons or having appointments, most of teacher-trainers perform their other tasks at home, i.e. curriculum development, lesson planning, checking papers, etc. When working at home, they have the freedom to structure their own schedules, i.e. having an appointment with the plumber during the day and compensating that time by working late at night.

While the execution of one’s tasks is largely performed on an individual basis, the teachers also work together through the knowledge centers that they belong to. OI’s expertise in the area of
teacher training in special education is coursed through the different knowledge centers (KC). There are 7 knowledge centers (Table 5-1). Membership is to a particular KC does not imply mutual exclusivity. It is possible that one instructor member of one KC is also part of another KC. This is because one instructor could have more than one area of expertise.

Table 5-1. Case B: Knowledge Centers at OI

<table>
<thead>
<tr>
<th>Knowledge Centers (KC)</th>
<th>No. of members</th>
</tr>
</thead>
<tbody>
<tr>
<td>KC Adaptive</td>
<td>50</td>
</tr>
<tr>
<td>KC Behavior</td>
<td>64</td>
</tr>
<tr>
<td>KC Learning problems</td>
<td>50</td>
</tr>
<tr>
<td>KC JRK (High Risk Youth)</td>
<td>28</td>
</tr>
<tr>
<td>KC Management</td>
<td>34</td>
</tr>
<tr>
<td>KC Regional Education Centers</td>
<td>16</td>
</tr>
<tr>
<td>KC VO</td>
<td>1</td>
</tr>
</tbody>
</table>

Collaborative activities take place in between, i.e. curriculum and instructional development, course evaluation, planning and coordination, etc. Most of these activities take place through the groupware system. The central office together with the competency centers which each educator belongs, also organize several offline face-to-face meetings to further stimulate collaborative activities. However, in most instances, the educators work alone. In some locations, an educator hardly sees her colleagues in the region nor is she familiar with them. But that does not mean that she gets the feeling that she does not know them or cannot be acquainted with them. The groupware system helps her overcome that obstacle.

5.1.3 Groupware application description

FCC is a COTS integrated groupware application for supporting collaboration, especially in the domain of education. It contains a range of built-in features for enabling collaboration such as email, shared workspaces, asynchronous chat, conferencing, individual and shared calendars, private and public directories. Further, the system has an accompanying programming environment for developing customized applications called FCC-RAD or Rapid Application Development.

5.1.4 Methods and protocols

The research for this case was also conducted using qualitative means: interviews, observations, online document analysis and software artifact inspection. This time, all of the interviews were recorded and transcribed. The duration of the research lasted for approximately 6 months. The interviewees who are distributed all over the country were likewise visited in their place of work, which is usually at home. In some occasions, due to poor location accessibility, the interviews took place at the main office.

One unique feature of the data gathering process is how interviews were coordinated through the system artifact itself. The system administrator, who was the main contact for this case, added the researcher to the system, i.e. to FCC itself. Through FCC, the users were contacted and approached for interview appointments. Gaining access to the system as a registered user enabled us to further gather observation about the system, its functionality as well as the dynamics of its use.

A total of 17 interviews were conducted among the users of the system. This figure also
includes both the system administrators and the regular users. One user who is also a system administrator but not part of OI was also interviewed as per recommendation of the main system administrator of FCC. The type and number of interviews that were conducted are shown in Table 5-2.

Table 5-2. Case B: Stakeholders and interview schedule

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Role</th>
<th>Total no. of users</th>
<th>Respondents interviewed</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCC Administrators</td>
<td>System sponsors</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Special education teacher-trainers</td>
<td>End-users</td>
<td>250+</td>
<td>14</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>255+</td>
<td>17</td>
</tr>
</tbody>
</table>

While the number of interviews conducted with the system users is comparatively small in relation to the actual numbers of possible system users, the number of interviews on the other hand serves a representative sample of the users. This is because a large number of the educators that work for the Institute are part-time employees. Therefore for these employees, system use is not really intensive although they are part of the system by having user accounts.

At the same time, the 14 interviewees were also selected strategically in order to get a good quality of data. Some of the interviews for example come from different specialization clusters within the Institute. Likewise, most of them have different levels of knowledge about the system and varying intensities of use. Some of them are very advanced in terms of using the system while some openly confess that they are having problems using the system.

The following online documents were referred to for analysis: the website of the institute, the FCC website, publicly available electronic documents placed on the system about the FCC, i.e., user guides, help, etc., and other online content that is accessible for all users.

Data generation is conducted according to the data gathering protocol (Section 3.3.4) specifically making use of the Level 2 Questions for individual cases, some of which were revised in Chapter 4 (Section 4.7.2).

5.2 Requirements at pre-implementation: fortuitous application discovery

This phases refers to the time when the user organization first got acquainted with the system up to its initial deployment. It is characterized by fortuitous application discovery. OI learned about FCC in a conference where a Swedish special education postgraduate school gave a demonstration of the software. The Swedish school had implemented FCC, which was then a Macintosh-based system, to support their teaching and learning processes. The presentation impressed the participants from OI, namely the ICT Manager and a few teacher-trainers. They were very keen about the software because it was Macintosh-compatible. At that time, OI had been using Macintosh machines.

Aside from this, OI was also very enthusiastic about using ICT to support their processes. This interest comes from the increased visibility and diffusion of ICT and internet technologies in the educational setting – a bandwagon that OI wants to jump into. There is a shared view in the educational sector in which the use of ICT is positively valued and highly encouraged. In fact, ICT use is being promoted as means of bringing about renewal and improvements in the quality of education. Government support and funding for ICT initiatives in schools were available and for which OI would like to make use of.

5.2.1 Requirements statements

We were able to identify five key initial requirements during this phase (Table 5-3). Similar to the first case study, this groupware implementation project also proceeded without a formal or semi-
formal requirements process. By this, we mean no structured protocol and methodologies were 
employed to elicit requirements i.e. user participation. When we conducted the research, we were 
not able to find any form of explicit requirements documents, i.e. project plans or proposals about 
the project. Our main source of information about requirements is the people involved in 
the implementation from the beginning. These were the IT Manager and the senior FCC 
administrator. Each of these requirements statements is elaborated in the following paragraphs.

PRE_BP_1. “The institute wants to participate in the renewal and modernization efforts in the 
educational sector through ICT.” This requirement results as a reaction to the developments in 
the external environment of the institute. At the national level, there are government initiatives 
towards modernization through ICT. These efforts are of course made known to the institute, it 
being a government institution. The ICT Manager and the FCC administrator cited the electronic 
government project (http://www.elo.nl) as one of the efforts in which the government’s ICT 
policy is laid down and gets implemented. Accordingly, the institute takes part in this initiative. 
We classified this requirement as business problem as it indicates a goal that the organization 
wants to achieve.

Table 5-3. Case B: Requirements at Pre-implementation

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BP_1</td>
<td>The institute wants to participate in the renewal and modernization efforts in the educational sector through ICT.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BP_2</td>
<td>The institute wants to implement an e-learning software.</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BP_3</td>
<td>IT Department and FCC administrators at OI. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.</td>
</tr>
<tr>
<td>4</td>
<td>PRE_SP_1</td>
<td>The software has to be an e-learning software.</td>
</tr>
<tr>
<td>5</td>
<td>PRE_SS_0_1</td>
<td>The system should run on Macintosh machines.</td>
</tr>
</tbody>
</table>

PRE_BP_2. “The institute wants to implement an e-learning software.” Implementing an e- 
learning software is the institute’s interpretation of the governmental project on educational 
modernization through ICT, which they would like to take part of. There is a general tendency 
among teachers and school administrators to think along the lines of didactics when it comes 
using ICT applications in education. This understanding is derived from the educational 
technology concept of computer-based training (CBT) and distance education, where learning is 
enabled through the use of electronic means such as the computer. In other words, ICT in schools 
is generally understood as e-learning. This notion was repeatedly expressed by the senior FCC 
administrator:

We want to have e-learning software in the institute. We saw how it was successfully 
implemented at the Swedish institute.

PRE_BP_3. “IT Department and FCC administrators at OI. Their goal is to innovate 
educational and administrative processes in the institute by finding and implementing ICT 
solutions.” With respect to stakeholder problem definitions, PRE_BP_3 is a specification of this 
for the IT Department of OI. This requirement specifies a concrete departmental goal which 
cencapsulates the main function of the entire department. At this stage, we can observe that the 
problem definition of only one stakeholder group is made known. This is because their 
requirements were the ones that were made explicit and had a direct impact on the acquisition of 
the system. This does not mean however that the other stakeholders group did not have any 
requirements at that time. However, during the interviews it was revealed that the teacher- 
trainers’ requirements were not elicited when FCC was acquired. They did not have a direct 
participation in the decision about the system. Their requirements were unstated and remained
tacit.

PRE_SP_1. "The software has to be an e-learning software." Before realizing what they actually need in terms of a software product, the institute already knew what kind of software they would like to have. In other words, they already have a software product idea in mind, which is that of an e-learning package. They have actually been in search for a COTS application that will serve such purpose. Therefore when the main proponents saw a product demonstration of FCC, they realized it is a kind of software that they want to address their needs: an online learning environment with support for distributed communication and collaboration. However, the more dominant product idea was the functionality for enabling online learning. Therefore the stakeholder view is also consistent with the developer view that FCC is an e-learning software.

PRE_SS_O_1. "The system should run on Macintosh machines." This requirement is already known even before the system was acquired. In fact, it was a given because the institute was using Macintosh machines and if ever they acquire a new system, it was preferred that the new system should be compatible with existing resources. This statement is classified under the software solution specification domain because this specification already gives an exact description of what operating system the software has to be compatible with.

5.2.2 Remarks

What we have observed about the requirements that we were able to recover at this stage is that these are limited and very general. The level of precision of the specification that is known with respect to the properties and characteristics of the software is only up to the software product concept and the constraint, i.e. PRE_BP_1, PRE_SP_1 and PRE_SS_O_1. Stakeholders have a general idea of what they want as a software product as a whole but with very vague notions of its specific purposes.

It is also observable that the proponents had tried to define their problem in terms of a software product concept. In other words, before structuring their problem, they already had a software product concept in mind, i.e. PRE_BP_1 is defined in terms of PRE_SP_1. We can also observe that the requirements of only one stakeholder group is represented or at least made explicit.

5.3 Requirements at early implementation: shifts in implementation strategy

The implementation of FCC at OI underwent a shift in strategy when it was officially acquired and initially deployed in the institute. This new episode in the implementation lifecycle, which we distinguish as early implementation, is the phase when FCC was acquired, deployed and used for the first time in the institute. This is the time when end-users begin to get acquainted with the system and begin to make use of its functionalities. Specifically, this succeeding phase is characterized by pragmatic decision-making, improvisation, and technological accommodation leading to a gradual diffusion of innovation. Likewise, as the very first time to break-in a system, glitches are expected to happen, all of which are indicators signalling that system implementation is actually a means of introducing change. In addition, these problems relating to system use during this phase also imply that requirements have changed. All of these were observed in this case.

This phase commences upon the acquisition of the Macintosh version of FCC and terminated by the time when First Class Gold version 7.0 was about to be implemented. This is when we entered the case study. The interval from the commencement of this phase up to its terminations spans a timeframe of approximately 5 years.
5.3.1 Requirements statements

The requirements statements gathered from this phase are enumerated in Table 5-4. As the compilation shows, there is a marked difference between the set of requirements from the previous phase to this current phase. This difference is evidenced by the increase in the number of requirements. With this observation, we can say that requirements get to be known or are made apparent in the context of actual implementation and eventual deployment. We discuss the recovered requirements from this case as follows.

**EARLY_BP_1.** “The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.” When faced with reality of executing the actual deployment of the system, the system proponents suddenly became aware of the goals and inconveniences encountered by teacher-trainers when carrying out their tasks. The most visible goal FCC proponents were able to identify as an improvement area is the way teacher-trainers gather and exchange instructional materials with each other. The senior FCC administrator recalls:

> Some of them [teacher-trainers] have to drive all the way to the main office to have their materials reproduced. They needed those materials for the following day to be distributed to their classes.

The other issue is the cost at which communications and task execution takes place. At that time, communications among teacher-trainers take place through the telephone and correspondence takes place via fax or through the post. Costs for these transactions are reimbursed by the institute. The proponents saw that this is one area which can be improved by the implementation of ICT.

Consequently, the awareness of these concerns prompted the IT department of OI to take a different course of intention with regards to FCC implementation. Driving this initiative is an improved appreciation of the possibilities of FCC as a groupware technology.

### Table 5-4. Case B: Requirements at Early Implementation

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_BP_1</td>
<td>The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_BS_1</td>
<td>FCC will replace the currently existing Pegasus email system.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SS_F_1</td>
<td>The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_Q_1</td>
<td>FCC should also run on a Windows-based operating system.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_BS_3</td>
<td>All new teacher-trainers at OI have 14 days to familiarize themselves with FCC.</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_SS_Q_2</td>
<td>FCC has to be learnable for a new user within 14 days.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_BP_2</td>
<td>The teacher-trainers want to improve their competencies. Specifically, they want to improve their knowledge and skills about FCC.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_BS_4</td>
<td>The IT department should also function as a helpdesk and provide technical assistance to users when they questions or problems with the software.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_F_2</td>
<td>The software should facilitate online communication between teacher-trainers and students.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_SS_Q_3</td>
<td>The software has to be in the Dutch language.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_SS_Q_4</td>
<td>A Dutch version of a user manual should be made available.</td>
</tr>
</tbody>
</table>
EARLY_SP_1. “The COTS software FCC is also communication and collaboration platform.” Instead of utilizing FCC immediately as e-learning software, i.e. when it is supposed to be used a didactic tool in the classroom, the proponents of FCC instead used it to address practical needs, i.e. EARLY_BP_1. This comes from a changed view or mental model of the system where they saw the possibilities of using it as a communications and collaboration platform. This opportunistic view of the technology is influenced by the exposure to and increased awareness of the difficulties teacher-trainers encounter when executing their tasks. The combination of this understanding of software possibilities and of the issues affecting the problem domain led to a series of actions that comprise a new set of requirements.

EARLY_BS_1. “FCC will replace the currently existing Pegasus email system.” With the new meaning and purpose assigned to the software, the first step they took in realizing the improved communication possibilities of the software was to replace the existing email system in the department. This plan was piloted to a small group of instructor-users who work often in the central office. In this pilot implementation, the users began to make use of the system for more complex and intensive communication tasks such as electronic document exchange and increased email exchanges among each other.

The positive acceptance and disposition to use FCC by the pilot users were facilitated by the proactive efforts expended by the IT department. The FCC administrators actively took part in using the system as secondary end-users. They created shared workspaces and discussions forums (called conferences in the FCC). They contributed content in each of these system facilities and communicated these through email to the users. In a way, the pilot users were bombarded with information and email such that they refer to and made use of the system quite intensively.

EARLY_BS_2. “Communication and exchange of materials among teacher-trainers should take place through FCC.” The success of the local and pilot implementation of FCC among the co-located teacher-trainers prompted the IT department to elevate the use of FCC at the department-wide level. This means all the geographically-distributed teacher-trainers have to now make use of FCC. Specifically, the IT department had prescribed the use of FCC as a communication and document exchange platform. The software is intended to improve the way communication and collaboration, which prior to FCC, take place through the traditional means: telephone and post. In a way, this requirement serves as a solution to an already identified business problem stated as EARLY_BP_1.

EARLY_SS_F_1. “The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.” Consequently, actions proceeding from the specification of business solutions in the form of EARLY_BS_1 and EARLY_BS_2 have implications on the functional and quality properties of the software solution itself. FCC as a communication and collaboration platform (expressed as EARLY_SP_1) must have a corresponding implementation property that it is available 24/7, and offering a convenient and an economical way exchanging messages and instructional artefacts.

EARLY_SS_Q_1. “FCC should also run on a Windows-based operating system.” As most of the teacher-trainers that work at home have PCs that run on Windows, propagating FCC to these users entail that the software must also be compatible with the Windows operating system. While it is possible for remote users to access FCC system services through a web-browser, which is platform-independent, the IT department chose to acquire the Windows-based package of FCC when it was made available for release. In this manner, home-based users can make use of FCC’s native client, instead of a web-browser. This decision was further enabled by the organization-wide home PC project in which employees of C3 that work at home are entitled to.

EARLY_BS_3. “All new teacher-trainers at OI have 14 days to familiarize themselves with FCC.” When the majority of the remote workers have been networked through FCC and its use is
already institutionalized, OI embarked on an official policy that all new teacher-trainers must be able to familiarize themselves with FCC. As most teacher-trainers work remotely and oftentimes alone, FCC serves as the enabling tool through which one instructor gets in contact with a colleague. One new employee we interviewed stated the following:

I was given a computer and a CD. I was told to install it and learn it in two weeks. When I met the area manager personally for the first time, the very first thing he asked me was, ‘are you already part of FCC?’. He said that another instructor lives in my area and he told me to search for her name and email her and introduce myself to her.

EARY_SS_Q_2. “FCC has to be learnable for a new user within 14 days.” That employees have 14 days to familiarize themselves with FCC, i.e. learn the basics of email, conferences or discussion boards, also means that it is a desirable quality attribute of FCC that it is learnable within 14 days. This is an example of a quality requirement.

EARY_BP_2. “The teacher-trainers want to improve their competencies. Specifically, they want to improve their knowledge and skills about FCC.” With the institutionalization of FCC as the default communication and collaboration at OI, the teacher-trainers felt an increased pressure on their part to learn the system. In fact, they have mixed reactions:

I find FCC a very convenient way of working. I can receive and respond to messages about my work at my own convenience. Whereas with the telephone, you have to answer it when it rings. – Donna, teacher-trainer

The thing with FCC is that, it goes on and on. Even on weekends you get messages. Late at nights too. – Nicole, teacher-trainer

Whether they perceive the use of FCC positively or negatively, the teacher-trainers felt the need to learn more about the software and to use it purposively and meaningfully. This is supported by the following user statement:

I have been using it for about year now since I joined the institute. I received the installation CD and learned it myself. I am quite sure I don’t know all the functions of the system and I am not sure whether the way I use it is the right way to do it. There may shortcuts, but of course I don’t know them. I actually asked the senior FCC administrator to organize a small seminar, you know like one afternoon where we all can come here [to the central office], and get a good tutorial. That would really be useful for all new employees. – Joe, teacher-trainer

EARY_BS_4. “The IT department should also function as a helpdesk and provide technical assistance to users when they questions or problems with the software.” As indicated in the discussion of EARY_BS_3, there were no training sessions or tutorials given to users. Users have to learn the software themselves. Therefore, whenever they encounter problems using FCC, they call the FCC administrators. It is usually the case that they directly call the senior FCC administrator who happens to be a very active promoter of FCC use. In this manner, next to administering and maintaining implementation, the IT department also functioned as a helpdesk. This action reflects a structural change in the organization and therefore signifies a business solution.

EARY_SS_F_2. “The software should facilitate online communication between teacher-trainers and students.” This requirement is a more specific formulation of the software product concept that software proponents had when they first saw FCC and regarded it as an e-learning system. This formulation is borne out of a more profound understanding the issues surrounding the teaching preparation processes as well as the possibilities offered by the technology. When they started using FCC, logistics issues in delivering instructions surfaced and came to be known. For example, teacher-trainers going all the way to the central office to have study materials
reproduced and bound in order bring them to the class for distribution the following day or two is an area of improvement.

EARLY_SS_Q_3. “The software has to be in the Dutch language.” Aside from being Macintosh-based, the original version of FCC that was implemented was in the English language. While this was not really a cause of concern for the users as they can work with the English version of the software, they would rather prefer to have the software in their own language. However, at this stage, the software remained to be in English version as a Dutch version is not yet available. This requirement was taken into serious consideration especially in the planning for future upgrades and acquisitions.

EARLY_SS_Q_4. “A Dutch version of a user manual should be made available.” While a user manual was available online, it was in English. The users preferred to have a manual written in Dutch. To satisfy this requirement, one of the FCC administrators translated a manual into Dutch and made it available through FCC.

5.3.2 Remarks

During this phase, there is an observable improvement in the requirements that were gathered. This is indicated by the increase in the number of recoverable requirements compared to the previous phase. Stakeholders developed a more profound sense of understanding and awareness of the requirements when they come in live interaction with the software. Because of this, the quality of the requirements that we have been able to identify has improved. This is evidenced by more informative requirements definitions. Specifically, we have seen a proliferation of requirements definitions in the domain of software solution specification. Functional and quality specifications were cited by stakeholders.

Critical in enabling this transformation are the proactive efforts taken by the people involved in the implementation – both system proponents and users. The positive and opportunist attitude towards technology held by the system proponents served as catalysts in the diffusion of innovation in the institute. The largely tech-savvy mindset of the system administrators enabled them to gain a more profound appreciation of the collaborative features of FCC. It enabled them to tune the software as an authentic solution to a practical problem faced by the employees. This mindset enabled them to shift from a strategic, externally oriented goal (PRE_BP_1 and PRE_BP_2) to an operational goal and pragmatic solution (EARLY_BP_1 and EARLY_BS_2). Likewise, this mindset was also made visible by the active utilization of FCC by the administrators themselves. They did this by regularly posting new information, news items, creating shared folders to upload utility softwares, i.e. Winzip, anti-virus, and using the email component to disseminate information. By actively using the system, they demonstrated to users new ways to make use of FCC, which further stirred and stimulated use.

The same computer-savvy mindset was also found in some teacher-trainers. Their enthusiasm and curiosity in using computers translated into an equally intense interest for FCC. In fact, users did not make clear discrimination between computer skills and software skills. To many of them, using a particular software application simply means using the computer to which they openly express they don’t have a good understanding of. They treat all software applications the same and do not make differentiations among them such that they don’t distinguish whether the difficulty they encounter is FCC-related or is due to other software or the internet connection. This affinity with computers and software was generally observed among the male teacher-trainers that were interviewed. They took initiatives. They experimented with FCC, i.e. trying out all features such as automatic reply messages when one goes on vacation. Such enthusiasm sparked other teacher-trainers’ interest in how FCC could further benefit them.

In the span of this phase, several organizational transformations also begin to take place. These transformations refer to structural changes induced primarily by behavioral adaptation and
coping. Individuals making use of FCC have evolved several behavioral mechanisms that accommodate, if not, reinforce the use and purpose of FCC. For example, requirement specification indicated by EARLY_BS_3 and the example quoted therein, reveals specific user behaviors that serve to underscore structural and organizational change. The area manager had already assimilated FCC into his daily work routine and was actively diffusing this innovation to a new colleague who was willing to accommodate and learn the technology as part of her new job.

As earlier noted, there is the remarkable articulation of desired functional as well as quality properties of the implemented software. This is reflected by the proliferation of requirements indicators in the software solution specification domain. This implies that as the software becomes more and more intertwined with the work processes and as users begin to get used with it, the requirements that surface are more precise and are software behavior driven. Prolonged exposure to the software induces an insight and an appreciation of the internal properties of software which users eventually learn to articulate as requirements.

5.4 Requirements at post deployment: from groupware to eduware

Post-deployment is going further than the 5 years after FCC was first deployed in the institute. This is the time when OI was already on its third cycle of software upgrade. It acquired version 7 of FCC, which was packaged as First Class Gold. This phase is characterized by an institutionalized system use which has led to significant changes in the department.

5.4.1 Requirements statements

At this advanced stage of implementation, the demands for the system have also changed leading to a new set of requirements. These requirements are identified in Table 5-5.

Table 5-5. Case B: Requirements at Post Deployment

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POST_BP_1</td>
<td>The teacher-trainers want to broaden their about the useful functions of the software in addition to the ones that they already know.</td>
</tr>
<tr>
<td>2</td>
<td>POST_BP_2</td>
<td>teacher-trainers would like to get to get to know other colleagues in a more personal manner than online.</td>
</tr>
<tr>
<td>3</td>
<td>POST_BP_3</td>
<td>Teachers would like to work more in teams when doing consulting for schools.</td>
</tr>
<tr>
<td>4</td>
<td>POST_SP_1</td>
<td>The system is the default communicator tool in the organization.</td>
</tr>
<tr>
<td>5</td>
<td>POST_SP_2</td>
<td>The software serves as a proxy for the virtual organization.</td>
</tr>
<tr>
<td>6</td>
<td>POST_SP_3</td>
<td>FCC supports e-learning and distance learning.</td>
</tr>
<tr>
<td>7</td>
<td>POST BS_1</td>
<td>All teacher-trainers should begin using FCC in communicating with their students.</td>
</tr>
<tr>
<td>8</td>
<td>POST BS_2</td>
<td>The delivery of instructional materials to students should be in electronic form and disseminated through FCC.</td>
</tr>
<tr>
<td>9</td>
<td>POST BS_4</td>
<td>FCC serves as a ‘closed’ system exclusively accessible only to OI employees.</td>
</tr>
<tr>
<td>10</td>
<td>POST BP_4</td>
<td>Users get the feeling that work never stops and that there is always a sense of urgency to reply to queries.</td>
</tr>
<tr>
<td>11</td>
<td>POST BP_5</td>
<td>Users do not want to be flooded with unnecessary information on their FCC start-up screen.</td>
</tr>
<tr>
<td>12</td>
<td>POST SS_Q_1</td>
<td>There should be a way to minimize the proliferation of icons on the start-up screen of FCC.</td>
</tr>
<tr>
<td>13</td>
<td>POST BS_3</td>
<td>New employees should be trained on how to use FCC.</td>
</tr>
<tr>
<td>14</td>
<td>POST BS_4</td>
<td>Apart training sessions should be given for advanced system functionalities and features, tips and tricks, as well as proper ways of using certain system functionalities.</td>
</tr>
<tr>
<td>15</td>
<td>POST BS_5</td>
<td>Users should organize and sort information on their screen in order to avoid information clutter.</td>
</tr>
<tr>
<td>Item</td>
<td>Code</td>
<td>Requirements statement</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>------------------------</td>
</tr>
<tr>
<td>16</td>
<td>POST_BS_6</td>
<td>Do nothing; continue using FCC despite the unpleasant experience of working with crowded electronic interface.</td>
</tr>
<tr>
<td>17</td>
<td>POST_BS_7</td>
<td>Teachers should formulate new instructional methods in the classroom to deal with increased workload, i.e. emails with students.</td>
</tr>
</tbody>
</table>

POST_BP_1. “The teacher-trainers want to broaden their about the useful functions of the software in addition to the ones that they already know.” As FCC use is institutionalized, teacher-trainers have become more open to express their need to broaden their knowledge of the advanced functions and features of FCC. They have been very vocal about this need. One instructor in an interview states the following:

Maybe I am not using all the possibilities of the system. I only use it to send email and to add attachments. Up to now, I still do not get it. What if I just wanted to send a portion of a document, and not the whole document, how do I do that? – Anke, teacher-trainer

POST_BP_2. “Teacher-trainers would like to get to know other colleagues in a more personal manner than online.” The users took the opportunity in the interviews to verbalize their latent need. Specifically, this is about the structure of work in the institute. In most instances, teacher-trainers work alone and to some, this is a new experience because prior to their appointment, they have been working in a traditional in schools. Because of this work arrangement, they sometimes feel lost in space and they miss the aspect of working in a co-located team.

POST_BP_3. “Teachers would like to work more in teams when doing consulting for schools.” This is a suggestion from one of the teachers and is an instantiation of EARLY_BP_2 as a more precise requirement.

POST_SP_1. “The system is the default communicator tool in the organization.” When asked to describe in her own words what FCC is for the organization, she replied, “It is the communicator. We use mainly to exchange messages with one another, with colleagues and with students as well.” This view is consistent with the way the software champions saw the system when they first implemented it (EARLY_SP_1).

POST_SP_2. “The software serves as a proxy for the virtual organization.” This is another metaphor used by several users when referring to the role of FCC at OI. One newly-employed instructor described FCC the following way:

When I see that FCC icon on my screen, that for me represents the institute.

POST_SP_3. “FCC supports e-learning and distance learning.” This software product concept is a reinstatement of the previous software product idea software proponents had when they first saw FCC (PRE_SP_1). With the institutionalized use of FCC, the IT department finally decided to implement the original intention with the software.

POST_BS_1. “All teacher-trainers should begin using FCC in communicating with their students.” This is an implementation of the reinstated software product idea POST_SP_1. This policy was just a formalization of an already existing practice in the institute. Some of the teacher-trainers have already began using FCC to organize their teaching activities and in communicating with students. Accordingly, this policy is supplemented by another policy in the form another business solution, POST_BS_2.

POST_BS_2. “The delivery of instructional materials to students should be in electronic form and disseminated through FCC.” This statement is a concrete specification of what teachers should do with FCC in relation to their teaching duties. It prescribes how they should change the way they deliver educational materials to students and in which form they should be given. With
regards to the latter, the institute had also embarked on a digitization project in which most of the study materials are to be converted in pdf format and sent to students by mail. Instead of receiving materials on the first day of school in the form of binders, students receive them electronically by email. This effort is an indication of operational process changes which imply organizational transformation.

POST_SP_4. “FCC serves as a closed system exclusively accessible only to OI employees.” The convenience of not having to remember a colleague’s email through the public directory feature of FCC had led the users to a mental model of FCC as a closed system. They have formed a rather consistent and shared view of FCC as an exclusive system, and thought that for other people outside OI, it is not possible to send email to or receive email from them. However, the system in fact does allow this. All they need to do is simply type the email address of the person they would like to send email to. Only because they cannot find the names of these people in the public shared directory, they thought it is not possible. As a consequence, most teachers maintain a second email address from free services such as Hotmail and other internet service providers. The following teachers’ comments give insight to this:

When I went to Malaysia for vacation last winter, I was able to access FCC through the web. I can read my email and get in touch with my work. However, I applied for a Hotmail account so that I can send email to my children. But I didn’t know that it is possible to do that via FCC. However, my children are not part of FCC, they are not in the system and they don’t have it.

I also have another email address. I use that to communicate with the clients. I do that because they are not on FCC and they do not have FCC.

Likewise, this mental model of a closed system is also shared by the system administrator. Before this research was undertaken, the system administrator suggested that it is better for the researcher to be part of FCC in order to gain access to the educators. An account was created for the researcher for the purpose of accessing and coordinating with the users.

POST_BP_4. “Users get the feeling that work never stops and that there is always a sense of urgency to reply to queries.” The features for supporting communication not only enabled faster communication among the educators but it also induced a continued sense of urgency. With colleagues responding quickly and promptly, some teacher-trainers felt the increased pressure of work.

I find the system rather stressful. It increases work pressure unnecessarily. It gives you the feeling that work does not stop.

POST_BP_5. “Users do not want to be flooded with unnecessary information on their FCC start-up screen.” Sustained long-term use with the software indicates that it can lead to several interaction issues. As users learn how to add new icons on the shared workspace of a team member, there was a sudden proliferation of icons in the start-up page of FCC. Some users found this irritating:

All those icons pollute my screen. I don’t need most of them. I don’t even know where they come from sometimes. I try to delete some of them, but I can’t delete some. Sometimes, I thought I have deleted it and then next time, they appear again.

From time to time I organize my desktop. You see I am member of more than one knowledge center and I am also part of the management. Therefore I have access to a lot of the shared conferences in these groups and they are all scattered on my screen.

POST_SS_Q_1. “There should be a way to minimize the proliferation of icons on the start-up screen of FCC.” This is the corresponding software solution specification to interaction issue
expressed in POST_BP_5. This is desirable functional and quality attributes ascribed to the currently being used version of FCC.

POST_BS_3 and POST_BS_4 (Table 5-5: Items 13 and 14) are requirements that originate as suggestions from end-users. Both suggestions are intended as means to address their desire to improve their knowledge and benefits in using FCC, which is reflected in POST_BP_1.

POST_BS_5 and POST_BS_6 (Table 5-5: Items 15 and 6) capture the reactions and actions taken by users when encountering interface issues with FCC. These statements relate to the interaction issue specified in POST_BP_5. Specifically, they are specifications of user coping mechanism when encountering difficulties with the software. One coping mechanism is by taking a pro-active role in which they try adapt themselves to the software, i.e. structuring the information on their screen so that it easier to navigate. Other users on the other hand, choose the passive role and do nothing: despite the unpleasant experience of working with a crowded interface, they continue to use FCC just to get the job done.

POST_BS_7. “Teachers should formulate new instructional methods in the classroom to deal with increased workload, i.e. emails with students.” Ever since the students have become part of the FCC, the dynamics teacher-student communication has changed. Students have become more proactive in communicating with their teacher-trainers. The software offered them a convenient communication medium that enabled a new learning behavior: students took advantage of FCC to email their teachers whenever they have questions. On the other hand, this was not appreciated by some of the teachers because it unnecessarily increases their workload. For one, the amount of emails they receive and reply to have increased. The worst situation is that teachers have to reply to the same kind question multiple times. Second, the software, especially its email facility induces students to become lazy learners.

In order to address this issue some teachers working together have decided to structure their teaching process and formulate teaching rules with students. For example, students were made to work in groups and if they have questions or clarifications, they were told not to send emails individually but rather as a group. The group has to first collect and synthesize all the individual questions and only one email will be sent to the teacher. The context of this requirement and is best described by one of teachers who re-organized her teaching method:

With the ELO project through FCC, there is an increased individualization of the learning process. There is a time savings benefit, as assignments can be conveniently placed online. For example, in the subject area that we are teaching, dyslexia, there is a theoretical module to that subject. We can already send right away to the student the readings and the assignment and they can work on later. On the other there is also increased work pressure on the part of the teachers. We end up spending more time in answering the students’ questions and in giving them feedback. And when they ask you questions, you have to also respond as soon as possible.

That is why what we are trying to do, a colleague and myself, is to organize the learning and teaching process. We would like to minimize as much as possible the exchange of messages between teachers and students. So, one idea is to have the students compile their questions as much as they can and they can send it to us at one time. We would like to avoid having to answer the same questions from 21 students 21 times. The other system we are trying to work on is to organize them into groups, and they have to send their questions to us per group. In that way, they already have discussed their questions and had avoided asking similar questions.

5.4.2 Remarks

At post-deployment, which is approximately five years down the road since FCC was first implemented, several remarkable features are observable. First, it highlights an intensive and institutionalized use of the software. FCC has evolved to become the de-facto communication
platform employees make use of to communicate, exchange messages and perform cooperative work. It has also become the first tool of choice when the need to initiate contact with a colleague or the institute arises. To this effect, software dependent behavior has been established which is reflected in the following user comments:

I actually use FCC much more than the telephone.

When I feel the need to communicate with someone, or I have a question or clarification, I go first to FCC before using the telephone. Actually, it is only when I needed information right away do I make use the telephone.

Second, this phase is also characterized by software-driven shift in business operating strategy. The widespread use of FCC paved the way for new ways of thinking about how to restructure and carry out the organization’s primary processes. This is indicated by the propagation of requirements in the business solution domain. The business solutions that were thought of during this phase were all influenced or are related to the use of the groupware. In other words, the constant exposure with positive results arising from the use of the software has led to intentions and initiatives to introduce organizational changes. When these changes are being planned, the software is always included in the agenda.

Third, in this phase a shift in software implementation directive has taken place: from a groupware application intended for internal communication into an online learning platform, thus eduware. As a recall, the latter has been the original directive when the software was first acquired and implemented (See Section 5.2). However, in the actual implementation and use of the software, it was not followed. Instead, Of formulated a different intention for the software when it was actually rolled out in the organization. It was implemented as groupware application to support backroom processes, i.e. teacher coordination and shared preparatory work processes surrounding the production and development of instruction. With this shift, what we can observe is a situation in which an unfulfilled or changed goal resurfaces at later phase when the organization has become ready to tackle it. The shift in business strategy influenced by the software is a characteristic this phase shares in the beginning, i.e. pre-implementation.

Instrumental in reinforcing all these structural and organizational transformation brought about by FCC use is the way users have evolved their mental models of the software. Users have come to develop a mental model of the system in which the software functionalities come to supplant organizational functions and facilities. For example, FCC has become a virtual meeting point – a space for socializing and orienteering in space (POST_SP_2) for employees who are working physically apart from each other. This role ascribed to the software has already been indicated in the previous phase and is further reinforced as FCC use becomes widespread and sustained.

The proliferation of requirements in the business solution domain provides indications that organizational changes are taking place. Specifically, these requirements are about making the situation and context outside of software suitable for software. They tell the many ways in which the organization is enforcing behavioral changes among its employees as far as using and embedding the software in their processes. Some of the requirements in this domain are statements of workarounds, i.e. other means of working with the software to overcome usability issues. In addition, some of these statements are from the organization itself, i.e. originating from the IT department, while some originate from the users themselves. This means that organizational changes results from both collective and individual and actions.

What is remarkable to observe from this phase about requirements change is that requirements can evolve without necessarily implicating changes to the software. This is implied by the minimal amount of requirement that were gathered in the software solution specification domain (see Tables 5-4 and 5-5). In this phase, requirements evolution took place more as a result of contextual changes, i.e. the situation outside the software, than of software change.
remained stable in the sense that while its quality properties demanded improvement from its users, its functional specifications remain unchanged.

5.5 Requirements evolution as within-domain change

Sections 5.5.1 through 5.5.4 discuss the evolution of each requirements domain through a compilation of the requirements statements according to phases.

5.5.1 Business problem domain evolution

Table 5-6 is a compilation of the requirements statements belonging to the business problem domain. Out of this compilation, the evolution of the business problem domain can be characterized as:

- Prominence at the start-up phase and at the post-deployment phase, and
- Shifting level of focus in terms of the content and stakeholders involved.

Table 5-6. Case B: Business problem statements compilation - all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BP_1</td>
<td>The institute wants to participate in the renewal and modernization efforts in the educational sector through ICT.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BP_2</td>
<td>The institute wants to implement an e-learning software.</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BP_3</td>
<td>IT Department and FCC administrators at OI. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_BP_1</td>
<td>The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_BP_2</td>
<td>The teacher-trainers want to improve their competencies. Specifically, they want to improve their knowledge and skills about FCC.</td>
</tr>
<tr>
<td>6</td>
<td>POST_BP_1</td>
<td>The teacher-trainers want to broaden their about the useful functions of the software in addition to the ones that they already know.</td>
</tr>
<tr>
<td>7</td>
<td>POST_BP_2</td>
<td>Teacher-trainers would like to get to know other colleagues in a more personal manner than online.</td>
</tr>
<tr>
<td>8</td>
<td>POST_BP_3</td>
<td>Teachers would like to work more in teams when doing consulting for schools.</td>
</tr>
<tr>
<td>9</td>
<td>POST_BP_4</td>
<td>Users get the feeling that work never stops and that there is always a sense of urgency to reply to queries.</td>
</tr>
<tr>
<td>10</td>
<td>POST_BP_5</td>
<td>Users do not want to be flooded with unnecessary information on their FCC start-up screen.</td>
</tr>
</tbody>
</table>

Business problem definitions are observed to be predominantly identified during the pre-implementation phase of the project and at its post-deployment. This pattern is accompanied by corresponding shifts in the attributes of these requirements over time, i.e. the level in which these problems are felt and their content. In the beginning, most of these requirements emanate from an awareness of a problem which is eventually expressed in the form of desires (See Table 5-6, PRE_BP.1 to PRE_BP_3). Usually these desires are stated at the strategic level by stakeholders and system sponsors. Business problems are least prominent at the onset of implementation when the software is initially deployed. However, at much later phase after a prolonged interaction with the software, new business problems arise. This set of business problems are different from the ones that were formulated in the previous phases in terms of their content and from which stakeholders they are attached to. At post-deployment, the problem definitions are statements of needs by system users who perform their tasks through the system.

Therefore the second characterization of business problem evolution in this case can be further described in the terms of the following patterns. Business problem definitions change over time from a strategic formulation towards needs specification at the executive level. In other words,
there is an evolution from top-level definitions to task-based formulations of business problems. This is accompanied by a changing stakeholder group that occupy the problem space. In the beginning, during pre-implementation and early implementation, problems are articulated solely by the system sponsors; in the later phase, the problem definitions come from the system users which in this case are the teachers. This kind of dynamics can be attributed to the structure and interplay of roles in a project setting: there are project proponents – those wanted the project and participate in its realization, and project beneficiaries – those in which the project meant for. In this case, the latter are the teachers. As their participation was not solicited in the beginning, the beneficiaries came later in the process and therefore their inputs which constitute part of the requirements become known afterwards.

5.5.2 Business solution domain evolution

The business solution domain is observed to have evolved in the following manner (Table 5-7). First, business solutions were not immediately thought of at the beginning when the intentions of acquiring a software product were known. Second, business solutions became more prominent in the later phases – early- and post deployment, when interactions with the software product became possible and more intense. In other words, there is an increase in number of business solutions identified over time.

Table 5-7. Case B: Business solution statements compilation - all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY BS_1</td>
<td>FCC will replace the currently existing Pegasus email system.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY BS_3</td>
<td>All new teacher-trainers at OI have 14 days to familiarize themselves with FCC.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY BS_4</td>
<td>The IT department should also function as a helpdesk and provide technical assistance to users when they questions or problems with the software.</td>
</tr>
<tr>
<td>5</td>
<td>POST BS_1</td>
<td>All teacher-trainers should begin using FCC in communicating with their students.</td>
</tr>
<tr>
<td>6</td>
<td>POST BS_2</td>
<td>The delivery of instructional materials to students should be in electronic form and disseminated through FCC.</td>
</tr>
<tr>
<td>7</td>
<td>POST BS_3</td>
<td>New employees should be trained on how to use FCC.</td>
</tr>
<tr>
<td>8</td>
<td>POST BS_4</td>
<td>Apart training sessions should be given for advanced system functionalities and features, tips and tricks, as well as proper ways of using certain system functionalities.</td>
</tr>
<tr>
<td>9</td>
<td>POST BS_5</td>
<td>Users should organize and sort information on their screen in order to avoid information clutter.</td>
</tr>
<tr>
<td>10</td>
<td>POST BS_6</td>
<td>Do nothing; continue using FCC despite the unpleasant experience of working with crowded electronic interface in order to get the job done.</td>
</tr>
<tr>
<td>11</td>
<td>POST BS_7</td>
<td>Teachers should formulate new instructional methods in the classroom to deal with increased workload, i.e. emails with students.</td>
</tr>
</tbody>
</table>

The business solutions that were identified in this case fall into three broad categories: organizational polices, training and self-adaptation. Looking closely at the content of these solutions, they are all oriented towards accommodating and promoting the use of software. Stating it another way, the scope of business solutions in the context of on-going implementation gets narrowed down over time. Consequently, as the scope is narrowed, the number of solutions being identified increases.

Similarly, the evolution of business solutions is also accompanied by the participation of the system users in the later stages. These stakeholders contribute to the identification and in fact unconscious execution of a solution. When they encounter difficulties, problems and failures while using the software, they find workarounds on how to get their tasks accomplished. Or,
some choose to deal with and endure the breakdown as they go on performing their tasks and eventually call for help when they cannot perform their tasks anymore (POST_BS_6). In this manner, the evolution of business solutions can be characterized not only by taking positive actions, but also non-actions. The latter are very weak or rather non-optimal forms of solution in the sense that they do not really contribute to a solution. However, they assist in addressing the problem although not completely solving it.

5.5.3 Software product concept domain evolution

The evolution of the software product idea in this particular case has a very interesting feature: it is marked by a shift in the original product idea towards a new product concept and later in the process, a return to the original product idea (Table 5-8). The change is from the original notion of an e-learning software towards a communication and collaboration platform; when the second product idea was successfully executed in the implementation of the software, the stakeholders, especially the software sponsors reverted back to the initial software product idea. The second concept of a communications and collaboration platform is further developed into a metaphor for the organization and helped promote the notion of a closed system among the users networked by the system (Table 5-8; POST_SP_2 and POST_SP_4). On the other hand, the initial software product concept that was not immediately executed in the implementation can be considered as a persistent requirement. The fulfilment of this persistent requirement at post-deployment which is conditional upon the successful execution of the alternative product concept leads us to the following conjecture: the accomplishment of an alternative product concept provides the opportunity to address persistent or unfulfilled requirements.

In addition to these observations, we can also add the following. The change in the software product idea from pre-implementation to post-deployment did not necessarily lead to a change in the software solution specification. The product specifications of FCC as COTS remained the same despite the evolution in metaphors used to describe the software. In other words, a given set of software solution specifications can have one or more software product idea that can be assigned to it over time. This finding confirms the notion of *affordances* (Norman, 1998, 1990; Gibson, 1977, 1979) in which the actionable components of a design product can be more than what it is intended based on how the user perceives the software to be. These perceptions of what can be done with the software need not necessarily correspond to way it is designed to be used.

<p>| Table 5-8. Case B: Software product concept statement compilation - all phases |
|-----------------------------|------------------------------------------------------------------|</p>
<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SP_1</td>
<td>The software has to be an e-learning software.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
</tr>
<tr>
<td>3</td>
<td>POST_SP_1</td>
<td>The system is the default communicator tool in the organization.</td>
</tr>
<tr>
<td>4</td>
<td>POST_SP_2</td>
<td>The software serves as a proxy for the virtual organization.</td>
</tr>
<tr>
<td>5</td>
<td>POST_SP_3</td>
<td>FCC supports e-learning and distance learning.</td>
</tr>
<tr>
<td>6</td>
<td>POST_SP_4</td>
<td>FCC serves as a ‘closed’ system exclusively accessible only to Ol employees.</td>
</tr>
</tbody>
</table>

5.5.4 Software solution specification domain evolution

As the most common form of requirements evolution, the change in requirements in this domain is the most prominent. It is the domain in which the most number requirements changes is observed to have taken place (Table 5-9).

The pattern of evolution in this domain is marked by an increased intensity in the number of
changes at the middle phase of the implementation; the pre-implementation and the post-deployment phases on the other hand have minimal requirements. The proliferation of this type of requirements in the early phases of the implementation can be attributed to the initial interactions users have with the software. This post-deployment phase marks the official beginning of software use in which users make interact with the system for the first time.

Among this type of requirements, quality specifications relating to interface design and usability issues comprise the most common request for change or improvement. On the other hand, request for additional functions or extensions of these were absent.

Table 5-9. Case B: Software solution specification statements compilation - all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SS_O_1</td>
<td>The system should run on Macintosh machines.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SS_F_1</td>
<td>The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SS_Q_1</td>
<td>FCC should also run on a Windows-based operating system.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SS_Q_2</td>
<td>FCC has to be learnable for a new user within 14 days.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SS_F_2</td>
<td>The software should facilitate online communication between teacher-trainers and students.</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_Q_3</td>
<td>The software has to be in the Dutch language.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_SS_Q_4</td>
<td>A Dutch version of a user manual should be made available.</td>
</tr>
<tr>
<td>8</td>
<td>POST_SS_Q_1</td>
<td>The system should have a means of filtering unnecessary information for the user or at least, it should direct users on how to filter information such that the proliferation of icons on the start-up screen of FCC is minimized.</td>
</tr>
</tbody>
</table>

5.5.5 Remarks

On the whole, the highlights of this within-domain evolution can be summarized as follows:

- There is marked increase in the number of requirements over time with the post-deployment phase as the most dynamic phase; approximately half of the gathered requirements statements are from this phase. This finding reaffirms the generally-held view that requirements for groupware applications are known later in the process.

- The business domain, i.e. business problem and business solution predominates the requirements space. This indicates that within-domain requirements evolution is mainly characterized by non-software modifications.

5.6 Requirements evolution: impact relations

In Chapter 4, we introduced the idea of *impact relations* as requirements change mechanisms in which a requirement in one domain gets updated or resolved by way of another requirement in another domain. The changed requirement is a way of *resolving* or addressing the issue raised by the requirements in the other domain. We have identified 7 types impact relations, which are also present in this case. The compilation of these impact relations can be found in Appendix C through I.

5.6.1 M1. Business problem resolved by business solution

There are 8 instances of this impact relation gathered in this case which span all the implementation phases (Appendix C: Case B). In some of the impact relations, we can see a close link between the statements in a way that one follows the other either logically or sequentially as a natural reaction. For example:
• PRE_BP_3. “IT Department including eventual FCC administrators. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.” is resolved by EARLY_BS_1. “FCC will replace the currently existing Pegasus email system.”

• POST_BP_1. “The teacher-trainers want to broaden their about the useful functions of the software in addition to the ones that they already know.” resolved by POST_BS_4. “Apart from training, sessions should be given for advanced system functionalities and features, tips and tricks, as well as proper ways of using certain system functionalities.”

5.6.2 M2. Business solution motivating a software product concept

We have identified 7 instances of this mechanism which are mostly observed after the initial implementation of the software (Appendix D: Case B). Basically, these are actions conceived in the user environment aimed to maximizing software benefits which lead to various mental models of the system.

• EARLY_BS_2. “Communication and exchange of materials among teacher-trainers should take place through FCC.” motivating POST_SP_2. “The software serves as a proxy for the virtual organization.”; and POST_SP_4. “FCC serves as a ‘closed’ system exclusively accessible only to OI employees.”

With the nature of work among the teacher trainers taking place virtually via FCC, the software system represents the virtual organization. It serves as a window to the organization that they work for at home. At the same time, this way of working leads to forming other ideas about the system as a closed system in which only those who have FCC accounts can communicate with each other.

5.6.3 M3. Software product concept realized by a chosen software solution specification

The realization of the software product concept is observed in 3 instances. The impact relations appear to be straightforward implementations of concepts into specifications, i.e. an e-learning software product concept justifies the functionality that supports communication between teacher-trainers and students (Appendix E: Case B, Item 17, PRE_SP_1 → EARLY_SS_F_2).

5.6.4 M4. Software solution specification supported by a business solution

This mechanism is an indicator of behavioral adaptations the software taking place. There are 5 indicators of these adaptations observed during the later phases of implementation – in the transition between early implementation up to post deployment (Appendix F: Case B).

• EARLY_SS_F_2. “The software should facilitate online communication between teacher-trainers and students.” supported by POST_BS_1. “All teacher-trainers should begin using FCC in communicating with their students.”

A software functionality that enables online communication between teachers and students can only be effective in actual use when its use is mandated (POST_BS_1).

• POST_SS_Q_1. “There should be a way to minimize the proliferation of icons on the start-up screen of FCC.” supported by POST_BS_5. “Users should organize and sort information on their screen in order to avoid information clutter.” and POST_BS_6. “Do nothing; continue using FCC despite the unpleasant experience of working with crowded electronic interface in order to get the job done.”

These instances of M4 impact relations are interesting examples of varying resolutions taken to address the issue of on-screen information clutter. Some users suggest that fellow colleagues
should sort their desktops while others choose to do nothing. It boils down to individual working styles and preferences, or to lack of knowledge about the software.

5.6.5 M5. Business solutions leading to improved software solution specifications

The effect of having operationally-specified business solutions leading to an improved quality of specifications is observed through this mechanism. There are 2 instances in the early implementation phase that depict this process (Appendix G: Case B). We cite one of them as follows:

• EARLY_BS_3. “All new teacher-trainers at OI have 14 days to familiarize themselves with FCC.” leading to EARLY_SS_Q_2. “FCC has to be learnable for a new user within 14 days”.

It makes sense that for an organizational work standard of learning software within 14 days the software should support this. Consequently, this provides a very specific requirement for the software vendor.

5.6.6 M6. Software solution specification leading to new business problem(s)

This impact mechanism is observed in the transition phase from the early implementation to post-deployment. Improved insight of a specification leading to new problem definitions is observed in three instances in this case. These impact relations basically embody the issues encountered in using the software. The statements EARLY_SS_F_1 and EARLY_SS_Q_2 are indications of the desired design properties of FCC which at the moment of use are problematic for users (POST_BP_4; POST_BP_1 and POST_BP_5). (See Appendix H: Case B.)

5.6.7 M7. Business problem resolved by (re)defining the software product concept

This impact relation establishes the link between the business problem domain and the software product concept domain. This relation is observed all throughout the phases of the implementation (Appendix I: Case B). The M7 instances in this case show how stakeholders have attempted to provide a software product concept to their problems.

• PRE_BP_1. “The institute wants to participate in the renewal and modernization efforts in the educational sector through ICT”; PRE_BP_2. “The institute wants to implement an e-learning software.”; and PRE_BP3. “IT Department including eventual FCC administrators. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.” resolved by PRE_SP_1. “The software has to be an e-learning software.”

• PRE_BP_3. “IT Department including eventual FCC administrators. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.” and EARLY_BP_1. “The way in which teacher-trainers develop, edit and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.” resolved by EARLY_SP_1. “The COTS software FCC is also communication and collaboration platform.”

These examples show how the problem articulations have been resolved by two software product concepts: an e-learning software (PRE_SP_1) and communication and collaboration tool (EARLY_SP_1). In a given problem or set of problems, a certain software product concept can be conceived to address the need.

5.6.8 Remarks

The many instances of the impact relations gathered in this case highlight the following findings:
• Three impact relations appear to be most common in this case:
  – M1. Business problem resolved by business solution
  – M2. Business solution motivating a software product concept
  – M7. Business problem resolved by (re)defining software product concept

• The software product concept domain is the domain that is most associated with change, i.e. it is the recipient of most change.
  – On the other hand, these change in the product concept did not always have lead to software change, i.e. not all changes in the software product domain have a corresponding change in the software solution specification domain. In most instances, the desired software functions and quality remained the same.
  – Stated in another way, the previous finding indicates that a given set of specifications can effect more than one software product concepts aside from that which the specifications are implementations of.

5.7 Discussion

Overall, the application of the conceptual framework in this second case study further confirms the framework's usefulness as a theoretical tool for learning about requirements evolution. More importantly, this case also helped improve our conceptual framework by introducing the idea of requirements change triggers as added properties of impact relations. This section discusses these new contributions, applies them to the previous case and formulates the latest version of our model.

5.7.1 Requirements change triggers: breakdowns and initiatives

Chapter 4 introduces impact relations as requirements change mechanisms in which a requirement in one domain gets updated or resolved in another domain. The relation exists because a felt sense of urgency or importance is attached to the resolution of the source requirement of the impact relation. The level of importance attributed to the resolution of the implicated requirement is determined by stakeholders’ actions: the fact they have taken a consequent action by way of articulating another requirement justifies the need for the resolution. A further examination of the impact relations and the circumstances in which they take place suggests that change come in two forms: reactive and pro-active impact relations. Consequently, we will refer to these changing requirements as reactive breakdowns and pro-active initiatives.

A requirement is considered a breakdown when it articulates a state in which a smoothly proceeding action is disturbed or interrupted. The goals of that action cannot achieved (Hettinga 2002; Miller 1995). A breakdown creates a situation of non-transparency and disengagement: when something regarded as usual suddenly becomes the object of attention. For example, when the computer is working perfectly for a data entry clerk, she is hardly aware of its existence. But when it breaks down, suddenly the tool as well as the process comes to attention. This is observed in this case in the later phases when the teachers took notice of the increased number of icons in the default screen of their FCC workspace. This instance is a breakdown because the awareness of the usability issue in the software’s interface has become a source of irritation for the teachers. Earlier this was not mentioned. This notion of a breakdown is consistent with that of Bardram (1998), for whom, in the context of work, a breakdown is a situation in which the flow of work literally breaks down either due to insufficient means of work or when the objective becomes unstable.

On the other hand, not all impact relations are breakdown triggered. Sometimes, the change results from pro-active decisions, namely, initiatives. A requirement as an initiative is a pro-active
intention meant to introduce change in a non-breakdown situation. It usually incorporates a solution agenda, which when compared to a breakdown as a trigger of change, an initiative has a lesser sense of urgency for action to be taken.

Using breakdowns and initiatives as added properties that articulate requirements change in impact relations, we can update our list impact relations to include these types of change triggers (see Appendices C through I). For each impact relation, we would like to discuss a few examples.

M1. Business problem resolved by business solution

This impact relation is a combination of breakdowns and initiatives. The beginning of the implementation are marked by initiatives (PRE_BP_1 to PRE_BP_3; Appendix C: Case B, Items 9 through 11). The business problem statements indicate the intentions to improve the situation and the business solution statements emphasize the use of ICT applications to introduce innovation. On the other hand, the rest of the M1 impact relations are breakdowns. For example the statement EARLY_BP_1 arises from an assessment of the distributed cooperative processes which are deemed to be sub-optimal and are prone to task breakdowns. When the classical modes of information exchange are not functioning, i.e. the fax is not working and there is no postal service on Sundays, there is a discontinuity or a breakdown in the flow of information. These were resolved by business solutions that specify what needs to be done, i.e. knowledge transfer sessions to fill in the gaps in knowledge of how to use the software better (Appendix C: Case B, Items 12 through 16).

M2. Business solution motivating a software product concept

M2 impact relations based on this case are usually initiatives. The business solution specified in the impact relation M2 is a precise specification of an initiative reinforced by an accommodating software product. For example, the generally agreed policy of having communications and cooperative work processes through FCC (EARLY BS_2; Appendix D: Case B, Item 7) prompts the mental model of a closed system.

M3. Software product concept realized by a software solution specification

This case shows that the realization of a software product concept into a concrete specification can be either be a breakdown or an initiative. It is an initiative when the product concept itself is purely focused on a software product as whole, i.e. PRE_SP_1. “The software has to be an e-learning software.” (Appendix E: Case B). On the other hand, the impact relation can also be a breakdown driven when the software product concept is already a a re-assignment of software purpose (EARLY_SP_1). The original product concept of an e-learning system is not pursued and this can be regarded as a breakdown.

M4. Software solution specification supported by business solutions

In this case study M4 impact relations are generally characterized by breakdowns (Appendix F: Case B). The statements reflect the observed regularity that interacting with the software leads to breakdowns. The implemented software needed support and stimulation through business solutions in order to make them meaningful for the user environment. Therefore, policies such as POST_BS_3: “New employees should be trained on how to use FCC.” are needed.

M5. Business solution leading to improved software solution specifications

The impact relation M5 (Appendix G: Case B) on the other hand is initiative-laden because the statements EARLY_BS_2. “Communication and exchange of materials among teacher-trainers should take place through FCC.” and EARLY_BS_3. “All new teacher-trainers at OI have 14
days to familiarize themselves with FCC.” are formulated as initiatives themselves. They are opportunity-seeking actions aimed at stimulating the integration of FCC into the processes of the institute. Accordingly these business solutions lead to the specification of more precise software functions like EARLY_SS_F_1. “The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.” and EARLY_SS_Q_2. “FCC has to be learnable for a new user within 14 days.” In turn these desired functions already happen to be incorporated in the version of the software that was implemented as part of its product specifications.

**M6. Software solution specifications leading to new business problems**

Like M4, the impact relation M6 is largely breakdown-driven. As the mechanism already implies, new business problems arise because there is a breakdown in software use, i.e. the software being always available gives the feeling that work never stops (i.e. EARLY_SS_F_1→POST_BP_4; Appendix H: Case B).

**M7. Business problem resolved by (re)defining software product concept**

This impact relation can either be a breakdown or an initiative. At pre-implementation, most M7 impact relations are initiatives (Appendix I: Case B). For example, while the statement PRE_BP_1. “The institute wants to participate in the renewal and modernization efforts in the educational sector through ICT.” is a reactive impulse, it is an organizational response to an external stimulus in which the external stimulus itself is also an initiative. Governmental efforts of stimulating educational renewal through ICT can be regarded as an attempt to introduce innovation. At the time when this policy was implemented, there were no known breakdown situations that could be identified as triggers for its formulation.

On other hand, M7 breakdowns are about users’ reactions when making use of the system does not work for them as they want it to be. The requirements involved in this impact relation are statements of discrepancies between how the users would like to the situation to be when interacting with the software and how it currently is (Appendix I: Case B, Items 7 through 9).

**5.7.2 Revisiting Case Study A: breakdowns and initiatives**

Since the concepts of breakdowns and initiatives were only formulated in this case to elaborate impact relations, we would like to go back and apply these concepts to the first case. The classification of the impact relation statements as breakdowns and initiatives for Case Study A are already reflected in impact relations listed in Appendices C through I. In a similar approach, we would like to briefly evaluate the different impact relations in case study A in terms of breakdowns and initiatives.

**M1. Business problem resolved by business solution**

M1 is instantiated in Case Study A largely as a breakdown but it can also be an initiative (Chapter 4; Appendix C: Case A). As an initiative, this impact relation manifests the pro-active effort taken by the system proponents (KCS), i.e. desires to have a community, which culminated in the decision to implement a software system. For example: PRE_BP_3. “The KCS - non life insurance aims to translate the organizational goal of integrating and unifying the different sub-companies by developing a community of non-life insurance experts.” resolved by PRE_BS_1. “The adoption of a Knowledge Management (KM) strategy is a means of community-building.” On the other hand, the breakdowns specify the issues the organization is facing in the beginning and deviations in the implementation plans and expectations.
M2. Business solution motivation a software product concept

This impact relation is largely characterized by initiatives (Chapter 4; Appendix D: Case A). It reflects the business solution taken and software product concept conceived to realize the overriding goals that motivate the implementation of KENNISNET (PRE_BP_3. “The KCS - non life insurance aims to translate the organizational goal of integrating and unifying the different sub-companies by developing a community of non-life insurance experts.”), which can be regarded as a non-breakdown situation. We can say that the statements PRE_BS_1 and EARLY_BS_1 that motivate product concepts such as PRE_SP_1, EARLY_SP_1 and EARLY_SP_2 are approaches that pro-actively focus on solutions with software in mind.

M3. Software product concept realized by a software solution specification

In turn, the formulation of software solution specifications to realize the conceived software product concepts are also instances of initiatives and breakdowns (Appendix C: Case A). In the beginning of the implementation, this process is initiative-driven. PRE_SP_1. “The system should be a knowledge management system for enabling knowledge exchange.” arises from an initiative. On the other and, the evolved mental model POST_SP_1. “KENNISNET should be in the form of an expert system.” is a reaction to the breakdown in software use. Perhaps, a different product concept offering a different purpose and functionality can change encourage system use.

M4. Software solution specification supported by business solutions

M4 according to Case A is motivated by a breakdown in which EARLY_SS_F_1. “The initial version of KENNISNET must be improved; it should have more functions and features.” is an all-encompassing formulation of the breakdowns that result from user interaction with the first version of the system. This issue has been productively resolved by the decision to include end-users in the design process (EARLY_BS_1; Appendix F: Case A).

M5. Business solution leading to improved software solution specifications

This impact relation can either be a breakdown or an initiative (Appendix G: Case A). It is an initiative in a way that the business solution is conceived as a result of conscious decisions aimed at introducing innovation and change in a situation (i.e. PRE BS_1: “The adoption of a Knowledge Management (KM) strategy is a means of community-building.”). On the other hand, the business solution statement EARLY BS_1: “The end-users should be involved in the re-design process in order to come up with more definite requirements.” is breakdown induced. This action results into a fruitful resolution in which more definitive software specifications were derived, i.e. a knowledge taxonomy for classifying knowledge items (EARLY_SS_F_2).

M6. Software solution specifications leading to new business problems

The instances of M6 impact relations gathered from Case Study A reflect the breakdowns encountered when software has been to put to use: it leads to breakdowns. The source of the breakdown is the design itself (EARLY_SS_F_2 to EARLY_SS_Q_12; Appendix H: Case A), which provides an early indication of the problems that will arise with respect to software use at post-deployment. The breakdown situation in this impact relation is that the updated version of KENNISNET is not generating that much enthusiasm as expected by the system sponsors.

M7. Business problem resolved by (re)defining software product concept

Case Study A presents the impact relation M7 as a breakdown induced requirements change (Appendix I: Case A). The problem statements POST_BP_1. “The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.” is resolved by
identifying other uses and purposes for the existing software (POST_SP_1 and POST_SP_2)

5.7.3 Conclusions and revised conceptual framework

So far, we have conducted two case studies of a failed implementation project and a successful implementation project, for which our framework appears to be helpful in documenting requirements change. Specifically, the second case led to improvements in our framework. In this case study, we have to come to enrich the concept of requirements change by adding the properties of breakdowns and initiatives to impact relations (See Section 5.7.1). These properties help us identify what triggers requirements change, which we have already raised in Chapter 4 as another level 2 case study question (see Section 4.7.2). Consequently, this added contribution is also used to review the impact relations from the previous case study and update its findings.

With these contributions, we have in effect achieved the goal set for this case study. Basically, we would like to apply the updated conceptual framework from Chapter 4 and establish its usefulness using a different set of conditions. Through the use of the case study questions (Level 2 Questions; Chapter 3.3.4, p. 48), we were able to gather requirements statement consistent with the requirements domains classification (Sections 5.2. through 5.4). Changes to requirements were articulated through within-domain evolution and the 7 impact relations for which we have also observed based on the requirements statements gathered (Sections 5.5 and 5.6). Each of these impact relations as requirements changes were further analyzed and enriched by adding the dimensions of breakdowns and initiatives (Section 5.7.2).

Key success factors

Addressing the question L2Q7 separately (Section 3.3.4 p. 48), which is not directly about conceptual framework, we have indentified the following factors that seem to contribute to groupware implementation success, which is defined in our study as sustained use of groupware with positive results.

Making use of the software to address organizational bottlenecks and operational breakdowns imposed by spatially distributed work

When FCC was introduced at OI, there were two purposes that can be fulfilled with it – one explicit and one implicit. The explicit purpose was the ICT modernization initiative. The implicit purpose was to provide support to teachers to overcome the difficulties imposed by distance in carrying out their tasks. Choosing to address the second purpose in the actual implementation of the system was crucial in establishing the usefulness of the groupware application. With this purpose, the software was able to bring about immediate improvements and concrete solutions to the work

Supporting use of software with organizational policy, particularly linking with reward systems, namely employment conditions

This makes the use of FCC motivating enough if not compulsory. In addition, this also limits the scope of exploration space, i.e. the teachers do not have to search for other communication tools because a product to be used is already identified.

Presence of tech-savvy users

This case also showcases the importance of people with widened mental frames in the process (Orlikowski, 1996; Bijker, 1986). Tech-savyness is a manifestation of this widened mental frame in the sense that a few of these teachers put forward ideas on how else FCC can be used for other tasks and processes within the Institute. They not only put forward the ideas and
suggestions but they also actively took part in exploring the feasibility of the option by conducting little experiments at home themselves. The widened mental frame of these users is about ICT generally, of which FCC is but an example. Therefore, the suggestions for ICT-based innovations by these users to the Institute was not limited to FCC; they also proposed the use of the other software applications in their work processes. For example, these tech-savvy teachers would like to implement the use of digital handouts in the form of Adobe Acrobat files to be sent via FCC to students before the start of the course. This is a change from the current way of working in which the teachers bring the course materials, i.e. 20 binders course materials in the first day of class.

**Revised conceptual framework**

With the completion of two case studies of opposing theoretical categories, i.e. implementation outcome, our conceptual framework has undergone a series of improvements since its initial formulation in Section 2.7. Its most updated version consists of the following propositions (REFP):

*Requirements domain matrix*

REFP1: Requirements can be defined in terms of domains: business problem, business solution, software product concept and software solution specification. Each of these domains is instantiated by requirements statements (Section 2.7.1; Chapter 3: Figure 3-1)

*Requirements evolution: within-domain evolution and impact relations*

REFP2: Requirements change is change in requirements from one temporal phase to another.

REFP3: *Within-domain evolution* is the change in any of the requirements domain from pre-implementation to post-deployment.

REFP4: *Impact relations* are dynamic requirements evolution mechanisms in which requirements change results from the update or improvement of one requirements statement from one domain into another domain. There are 7 known impact relation mechanisms.

*Impact relations: breakdowns and initiatives*

REFP5: *Impact relations* are resolutions of breakdowns and initiatives. Breakdowns and initiatives are requirements statements that articulate an issue which prompts stakeholders to take or propose actions as means of resolution or closure.

A visualization of this updated framework will be given in the next chapter, which we will conclude as the final model and use in the last two case studies.
A CONCEPTUAL FRAMEWORK OF REQUIREMENTS EVOLUTION

In Chapters 4 and 5, we applied the conceptual framework we developed in Chapter 2 in two case studies. This proved to be a useful exercise because we were able to gather data from the cases and establish results. More importantly, we were able to move forward and improve the conceptual framework. In this chapter, we will present the updates made on the conceptual framework (Sections 6.1 through 6.3) and the plans for additional case studies (Sections 6.3 and 6.4).

6.1 Requirements evolution as within-domain evolution

In Case A (Chapter 4), we introduced the concept of within-domain evolution, which is a technical refinement of our initial definition of requirements as the change in the state or contents of the requirements domain over time (Sections 2.7.2 and 3.3.1; Figure 2-6). Within-domain evolution is repeated in Figure 6-1 to show requirements evolution as the change in each requirements domain over different temporal phases. On the hand, the concept of requirements domains as whole as well as its components did not change.

![Figure 6-1. Requirements evolution as within-domain evolution](image-url)
6.2 Requirements evolution as *impact relations*

From both cases, we further expanded our idea of requirements evolution as interaction between domains. For this, we introduced the concept of *impact relations*. An impact relation is a connection between requirements statements belonging in different domains. Requirements evolution is characterized as a dynamic process resulting from an impact relation: a requirement in one domain gets updated or resolved in another domain. Beginning from Case A, we were able to identify 7 types of impact relations (Figure 6-2; Table 6.1)

![Figure 6-2. Requirements evolution as impact relations](image)

**Table 6.1. Impact relation codes and definitions**

<table>
<thead>
<tr>
<th>Code</th>
<th>Impact Relation Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Business problem resolved by business solution</td>
</tr>
<tr>
<td>M2</td>
<td>Business solution motivating a software product concept</td>
</tr>
<tr>
<td>M3</td>
<td>Software product concept realized by software solution specification</td>
</tr>
<tr>
<td>M4</td>
<td>Software solution specification supported by business solution</td>
</tr>
<tr>
<td>M5</td>
<td>Business solution leading to improved software solution specifications</td>
</tr>
<tr>
<td>M6</td>
<td>Software solution specification leading to new business problem</td>
</tr>
<tr>
<td>M7</td>
<td>Business problem resolved by (re)defining software product concept</td>
</tr>
</tbody>
</table>

6.3 Breakdowns and initiatives

In line with impact relations, we also introduced breakdowns and initiatives as added properties of impact relations (Section 5.7.1). They are requirements that trigger change. Usually, a requirement is an articulation of an issue in the practical world. An impact relation is brought about the resolution of breakdowns and initiatives. A breakdown is a break or a disruption in a smoothly proceeding process. It is captured in a requirement statement that tells about what broke down in the situation.

On the other hand, an initiative is a non-breakdown situation. It is pro-active action taken to introduce changes or improvements in situation which is stable, i.e. users are not complaining, work is proceeding as usual and the software goes into the background. Just like breakdowns, initiatives are also expressed as requirements statements.

Therefore, requirements evolution is the resolution of a breakdown or an initiative resulting into an impact relation.
6.4 Additional case studies

On the basis of this updated framework, we can proceed with the conduct of two more case studies. Additional studies are warranted given the extent of updates we made from the basic or initial framework towards this interim model (Sections 6.1 and 6.2). The question remains open whether there are still updates to the framework that can be added based on new empirical data. In carrying out the additional cases, we formulated the following plan and questions.

6.4.1 Case types: polar examples of discontinued vs. continuing use of groupware

The same case study design will be followed (Section 3.3.3) in which 2 similar cases and 2 contrasting cases are studied. This is a criterion for their selection. Case Study A is a failed implementation and Case Study B is a successful implementation in the sense that it is an instance of continuing use of groupware. Therefore for the next two cases, we need a pair of failed and successful groupware implementation projects.

6.4.2 Updated research questions

In the light of the updates in the framework, additional research questions arise. The following is the complete overview of the updated research and relevant data gathering protocol questions (See Sections 1.3.3, 3.3.3 and 4.7.2). These will be applied in the next two cases and in the concluding analysis of this research.

Research question: level 4 question

Central Question: What are the requirements change mechanisms in groupware implementation and use?

Individual case study questions: level 2 questions

L2Q1 What are the requirements (broadly defined in terms of our proposed requirements domains matrix) that motivate the groupware implementation in this case?

L2Q2. Who are the relevant stakeholders?

L2Q3. What is the groupware application in use?

L2Q4. What are the changes in each domain across the phases (within-domain change)?

L2Q5. What are the impact relations in this case?

L2Q6. What are the breakdowns and initiatives in the impact relations?

L2Q7. Are there additional impact relations that can be discovered in this case aside from the ones already known?

L2Q8. What lessons can be learned from this groupware implementation either as a failed or successful implementation?

Pattern-seeking questions across multiple cases: level 3 question

L3Q1 What are the common patterns of change among the cases?

L3Q2. What can be learned from the aggregated findings across cases about requirements evolution as a phenomenon?

L3Q3. What can be concluded about the conceptual framework and its application in a multiple case study research?
Normative questions about policy recommendations and conclusions: level 5 question

L5Q1 On the basis of the study conclusions, what actions or guidelines can be given to organizations hosting groupware that will help them manage the process more efficiently and purposefully?

L5Q2 What advice or heuristics can be given to designers of collaboration technologies such that the uptake and use of these technologies are indeed supportive of human activities and purposes?
Chapter 6 defines the definitive conceptual framework for this research, to be validated by two more case studies. This chapter analyzes Case Study C, which is about the long-term use of a Web-based course and learning content management system. It is an example of a successful implementation of groupware because of its prolonged use.

Section 7.1 provides the background for the case study. Sections 7.2 through 7.4 present the case and the requirements according to the temporal scheme. The focal points of this chapter are Sections 7.5 and 7.6 in which the framework concerning requirements evolution is confronted with the data from the case. Section 7.7 concludes this chapter with a discussion of the contributions of this case and lessons learned.

7.1 Case background

Case Study C is about the implementation of TeleTOP, a web-based course management system. While its implementation is at a university-wide level, the case study focused on smaller scale implementation, i.e. departmental level. The implementation features a long-term use of groupware for which this case is considered successful.

7.1.1 Implementation setting: Department of Industrial Design Engineering

The Department of Industrial Design Engineering was established when the program Industrial Design was established as a full study program rather than a specialization within the Faculty of Mechanical Engineering. With the larger Mechanical Engineering Faculty (MEF) already making use of TeleTOP it seemed to be a natural course of events for the spin-off department to follow suit and implement the system.

Within the university, TeleTOP is the default course management system that is being used in all its faculties and departments. It was developed within the Faculty of Educational Technology (FET). Its implementation at the university-wide level took place incrementally, i.e. not rolled out at the same time but one faculty at a time. The system was already in its fourth year of use in MEF when its use was extended to the newly-formed department. Version 5.0 of the software is evaluated in this study. This indicates that TeleTOP had undergone already a series of design
changes ever since its initial conception and it continues to do so in its commercial releases.

7.1.2 Stakeholders

The main stakeholders within the scope of this case study are the system users that perform the roles of teachers and students. As new program of study, students can be further grouped into new students – first year students at the time of the study and old students – students on their second year or higher. The system proponent is the university, which within the context of the implementation setting falls outside our scope, but its mandate and influence to the implementation is represented through the Faculty administration will be reflected in the analysis.

7.1.3 Groupware Application Description: TeleTOP development background

TeleTOP stands for Tele-learning Toegepaste Onderwijskunde Project which is an electronic learning environment initiative at the Faculty of Educational Technology. Its development started as early as 1995 spearheaded by a team of a professor and six educational technology experts. Similar to Case Study A, the development of this groupware is an in-house effort originating from the ground up. Tele-learning (Collis, 1999) was a term coined to refer to the utilization of ICT technology in enhancing the learning process. Back then, the term e-learning was not yet a major hype. Specifically, the project made use of WWW technology implemented through Lotus Notes Domino.

As an educational tool, TeleTOP is being used a course management system supporting the work of teachers and students. Each subject has its own website in the system and its content are managed and provided by the respective teachers responsible for the course. It assists teachers in scheduling the course sessions, uploading hand-outs and other teaching materials. Students on the hand are supported by the tool in gaining an overview of their subjects they taking for the relevant semester. For each subject, they can access and view additional information such as the schedule of lectures and tutorials. They can also download the slides and handouts for each when these are made available by the To provide an idea of its interface and how the different user roles are distinguished in terms views, several screenshots are shown in Figures 7-1 to 7-4. Furthermore, the known specifications of TeleTOP as a finished product are listed in Table 7-3 (see 7.3 Requirements at pre-implementation).

![Figure 7-1. Teacher role view. The functions and services of TeleTOP are shown on the left hand side menu. Most, if not all functions are shown for the teacher role. The Roster (course schedule) is shown on the main frame. The icon on the top left side menu representing a small hand with a pen is the edit function. It is possible to customize the side menu items.](image-url)
7.1.4 Methods and Protocols

Qualitative approaches towards data gathering applied in this case include document analysis, individual and group interviews as well as software artefact inspection.

For Case Study C, the interviews were conducted among two major roles known to the system: teacher and students. A total of 9 interview sessions were conducted: 6 individual interviews with teachers and 3 group interviews. The group interviews consisted of different year-level students. Focused group discussions were also conducted in two groups of first year students who were involved in a project-based course. For older year students, the study group S.G. Daedalus was approached and they provided older year students: 2 second year students and 3rd year students. The schedule of the number of interviews conducted is shown in Table 7-1.
Table 7-1. Case C. Stakeholders and interviews schedule

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Role</th>
<th>Individual Interviews</th>
<th>Group Interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teachers</td>
<td>End-users</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Course Coordinator</td>
<td>System sponsor</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Students</td>
<td>End-users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st year students – Group A</td>
<td></td>
<td></td>
<td>3 (1st session)</td>
</tr>
<tr>
<td>1st year students – Group B</td>
<td></td>
<td></td>
<td>3 (2nd session)</td>
</tr>
<tr>
<td>Higher year students (S.G.Daedalus)</td>
<td></td>
<td></td>
<td>4 (3rd session)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>

The relevant case study questions updated in Chapter 6 (see Section 6.4.2) are applied in this case. Likewise, our improved conceptual framework of requirements evolution provides the structure in the presentation and analysis of case data.

7.2 Requirements at Pre-Implementation: following suit

The pre-implementation phase of TeleTOP can be characterized as following suit. As a spin-off department from the Faculty of Mechanical Engineering with all facilities in place, the Department of Industrial Design Engineering merely took over what’s available from the bigger organization. These include project labs, teachers, students, facilities and resources, among which include TeleTOP. In fact, some of the teachers that teach in the Industrial Design program also teach in the Mechanical Engineering program. Therefore, with respect to new requirements motivating the implementation of TeleTOP in the school, they are actually limited if practically non-existent.

Aside from the already existing TeleTOP infrastructure in the faculty, the accession to TeleTOP was made a lot easier and faster by the university-wide mandate of using TeleTOP in all faculties and programs. As mentioned earlier, TeleTOP was incrementally on its way to becoming the default digital learning environment of the entire university. By the time the school of Industrial Design was established, and with it using TeleTOP, the coverage of implementation has reached 100%.

7.2.1 Requirements statements

Tables 7-2 and 7-4 list the requirements statements identified in this phase. Specifically, Tables 7-3 and 7-4 are listings of requirements statement that more or less represent the product specifications of TeleTOP. As a recall, in this implementation, TeleTOP was already an existing system, for which the department in this study was simply continuing. The functional (Table 7-3) and quality properties (Table 7-4) of the system are more or less known.

Leading the requirements list in Table 7-2 is a software product concept PRE_SP_1 which specifies the prevailing mental model about TeleTOP. It represents the product idea the designers have propagated about the software as well as the general perception of its proponents (Collis et al., 1999). This way, it is top-down encompassing requirement which originates at the university level that is also relevant to the school.

Going back several years before the implementation of TeleTOP at the Department of Industrial and extending the scope outside the mechanical engineering faculty, there is a lot of background information that shed light about the requirements that motivate the development of TeleTOP. Therefore, the software product concept that we have identified in this period, PRE_SP_1 is already an evolved requirement. The system is meant as a classroom support tool to enhance the teaching and learning process by making these more flexible and efficient. As a tele-
learning tool, the underlying philosophy imbued by the developers in the system’s development is purely didactics. However, this software concept had shifted by the time the implementation was put on the wider scale of the university with many faculties of differing domains.

Table 7-2. Case C. Requirements at Pre-Implementation

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SP_1</td>
<td>TeleTOP is the tele-learning tool; it provides support in the teaching and learning process.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BS_1</td>
<td>TeleTOP is the mandated default learning environment in the university; all faculties and schools should make use of it.</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BS_2</td>
<td>TeleTOP is already an existing resource for supporting educational tasks in the university and faculty. The school of industrial design engineering must make use of it.</td>
</tr>
</tbody>
</table>

PRE_BS_1. “TeleTOP is the mandated default learning environment in the university; all faculties and schools should make use of it.” As one of the early development and implementations of ICT in education, TeleTOP was a pioneering effort: it was the first WWW-based tele-learning tool developed in-house in the university, and the first of its kind in the country. This was also one of the reasons why it gained the needed administrative support to have it implemented throughout the university. Implementing it is consistent with the vision of becoming a leading edge, pioneering digital university. Software use did not spread instantaneously. Deployment took place incrementally following a series of pilot implementations, with several software enhancements made after each pilot. Piloting was first conducted within the faculty and this spread out to other faculties such as public administration, computer science and mechanical engineering. The cooperation of other faculties was obtained by the very pro-active efforts taken by the development, in particular its head professor in promoting and increasing the software’s visibility not only within the campus but within the academic community as well. Through the development group’s efforts, the project was able to obtain and endorsement from the University board as the software tool that would bring to fruition its vision of innovating education in the campus. Eventually, TeleTOP has become the default digital teaching and learning environment for all courses taught at the university. Consequently, a support organization has been set-up for providing technical and user support as well as helpdesk services. This is support unit is the ITBE, which is mainly responsible for IT and library services for the whole campus. It was organized that for every faculty, a TeleTOP expert / contact person was available. Meanwhile, the continuing design and development of later versions of remained within the original development team. Therefore, the statement PRE_BS_1 captures the discussion in previous paragraphs about the surrounding context that prompted the use of TeleTOP in school. This requirement is in the business solution domain because it represents an organizational policy that seeks to introduce innovation to the whole university. In turn, this makes the requirement initiative-laden because there was no explicit indication of a problem, or more specifically of a breakdown situation the policy is intended to address.

PRE_BS_2. “TeleTOP is already an existing resource for supporting educational tasks in the university and faculty. The school of industrial design engineering must make use of it.” At the department level, this statement is a follow through of the earlier statement PRE_BS_1. Knowing that the tool is available for use and it has the mandate from the university board, and even though they have the autonomy choose whatever software tool they would like to implement, the administrative board of the school of industrial engineering has decided to proceed with using TeleTOP. According to the the program and education manager:

We have decided that since TeleTOP is now available, we made a conscious decision, that okay, from now on we have to do something with it. We have to use it.

Being already an operational finished product before the Department of Industrial Design
Engineering was established, the product specifications of TeleTOP are already known. These are identified in the software solution specifications PRE_SS_F_1 to PRE_SS_Q_2 in Tables 7-3 and 7-4. Correspondingly, these descriptions are the specifications that bring about the software product concept PRE_SP_1.

### Table 7-3. Case C: Requirements at Pre-Implementation - Functional Product Specifications

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Product specification</th>
</tr>
</thead>
</table>
| 1    | PRE_SS_F_1   | [ProdDesc] TeleTOP is a role-based web application; it recognizes two types of roles: teachers and students.  
- Access and to software functions are authorized on the basis of these roles. For example:  
  - Teachers have edit roles; they can create content  
  - Students usually have read roles, which is not limited to viewing content but also extends to uploading and downloading documents.  
  - Account name and password is needed in order to access system services. |
| 2    | PRE_SS_F_2   | [ProdDesc] Course information, registration and administration  
- Course information. TeleTOP allows teachers to provide general information about the course they teach (on a subject per subject basis). This include structured information such as allows teachers to provide general information about the course they teach such as course description, course goals, teachers, textbooks, and evaluation and format  
- Course registration. Students enrolled in a course are not automatically added registered TeleTOP. They also need to register for the TeleTOP site of the course in TeleTOP itself |
| 3    | PRE_SS_F_3   | [ProdDesc] Course planning, logistics and study support  
- Roster. Through the Roster, TeleTOP provides a schedule for organizing and structuring lessons on a session per session basis (see Fig. 7-1). Teachers can provide a program of lessons, learning activities, learning materials and exercise.  
- Digital lecture slides and handouts. TeleTOP has a facility for making slides, presentations and handouts available online. Teachers can upload them several areas: in the roster, archive and presentations. Likewise, students can also download the slides.  
- Assignments and exercises. Teachers can also upload the specification of assignment or exercises in TeleTOP via the Roster. In the same manner, students can also submit their assignments through TeleTOP. Presentations. The function represented by this button enables teachers to upload all kinds of presentations used in class, which not only include those given by the teacher but also those made by the students. |
| 4    | PRE_SS_F_4   | [ProdDesc] Off classroom communication  
- News. TeleTOP has a news section in which teachers can put announcements, updates and other information pertinent to the course that is otherwise not communicated in the classroom or needs to be emphasized outside of regular class hours. For example, a class will not push through or that the session is moved to a different classroom.  
- Emails and Groups. By registering to the TeleTOP site of the course, the names and email address of the students are captured in the system. They are shown in the Email/Group section. Through these, teachers can create student groups, i.e. like when they create groups for projects and exercises as a didactical approach. |
| 5    | PRE_SS_F_5   | [ProdDesc] Interactive online communication  
- TeleTOP also has a facility for interactive multidirectional communication (between students themselves and between students and teachers)  
  - Discussion. The discussion facility is mainly for students to discuss about the course. It functions as threaded bulletin board in which messages can be posted, instantiated by a topic and responses can be made to. Teachers can view the discussion.  
  - Question and Answer. This facility allows students to ask questions to the teacher and to classmates outside the classroom. In return, the teacher(s) and other students who have answers to the question can reply back. |
| 6    | PRE_SS_F_6   | [ProdDesc] Collaboration  
- The following functions are available for those with teacher roles, which are left for discretionary use. These functions are systems services that are on reserve, which can be set-up as additional utility for the users (see Figure 7-2)  
  - Glossary. Teachers can create a list of terms that are relevant for the course.  
  - Weblinks. Teachers as well as students (depending on the authorization set by the teacher) can upload interesting websites that are relevant or will be used in the course.  
  - Archive. This function allows teachers to upload all the digital files and software that is used or will be used in the course.  
  - Publications. This function is intended for uploading and downloading relevant articles and literature for the course.  
  - HTML Pages. This function is for setting up a central place for collecting HTML pages.  
  - Quiz. With the QMP integration, it is possible for teachers to set-up short quizzes.  
  - Poll. Through this function, teachers can set-up short opinion surveys in which students can vote.  
  - QMP Assessments. TeleTOP has a the facility for making online examinations using the COTS exam package Question Mark Perception (QMP).  
  - Administration. This function is for administering the different parts of the TeleTOP site, i.e. whether one function such as poll should be made available or not (see Fig. 6-3)  
  - Search. This function allows course-wide search. |
Table 7-4. Case C. Requirements at Pre-Implementation - Quality Product Specifications

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SS_Q_1</td>
<td>[ProdDesc] TeleTOP is available in both English and Dutch versions.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_SS_Q_2</td>
<td>[ProdDesc] TeleTOP has a text-based manual available.</td>
</tr>
</tbody>
</table>

7.2.2 Remarks

There are also very few requirements statements that can be derived from this implementation. This is due to the nature of the implementation itself. The department is simply continuing a much larger implementation strategy coming from the university board. Therefore, there is practically no need to specify requirements for a system is already known and assumed to be implemented. On the other hand, with the software product already known, the software solution specification domain is filled in with functional and quality product descriptions. In our matrix definition of the requirements, in the context of implementation, it is assumed that the software specification statements are consistent or compatible with the software product description.

7.3 Requirements at early implementation: propagating TeleTOP use and the fire incident

While introduction and subsequent use of TeleTOP at the department marked a complete university-wide deployment of TeleTOP, it did not necessarily mean that 100% of the intended users were using the software. Rather it meant that software was available for use for all intended users. This applies to TeleTOP at this phase of implementation.

Encouraging the teachers to make use of it in the Department of Industrial Design Engineering was a major concern at the beginning of the implementation. This is an already known problem. It is a problem that actually belongs to the central body, namely in the university that manages the deployment and maintenance of TeleTOP. On the other hand, it is a problem that is passed on to the educational bureau of which action and cooperation from them is expected. Likewise proactively, the ITBE has placed different contact persons and experts in each faculty to provide support and oversee the implementation and use of TeleTOP. For the department, there is also one such person. This person coordinated with the IT helpdesk of the faculty in assisting users in using TeleTOP, i.e. answering questions, providing personal assistance, showing demos and giving presentations and making available references online and offline for users to use. One of the professors interviewed said the following:

I usually do the tasks myself – of putting the courses there [in TeleTOP], the layout and others. However, if I want to do something else, then I call Lisa Gomers [TeleTOP contact person]. I talk only to her and when I have questions, I ask her.

An important event that took place during this period was the fire incident in the university. The building hosting the central computing facility was destroyed by fire. This led to the breakdown of all computing services including TeleTOP in the whole university. While this incident created a disruption in activities, in most instances, the operations in the university proceeded as usual but not smoothly. In the school of Industrial Engineering, the teachers were quick to adapt to the situation. One of the teachers remarked about the incident:

My TeleTOP site is that physical bulletin board over there [referring to the tack board outside his door]. I temporarily made use of that facility while TeleTOP was not yet on air.
Whereas in usual occasions, students and teachers communicate with each other electronically, under these extreme circumstances, they resorted to face-to-face communication. It is the case that the initiative to communicate is either taken by the students and teachers themselves. Students would directly go to their teachers rooms and teachers would look for the students in the lab or in the canteen.

### 7.3.1 Requirements statements

Given the background of the major events in phase, the requirements gathered during the early context of the implementation are listed in Table 7-5. The statements gathered span all the four domains of requirements.

*Table 7-5. Case C. Requirements at Early Implementation*

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_BS_1</td>
<td>There should be a TeleTOP expert available for school the of Industrial Design Engineering to oversee the system deployment</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_BS_2</td>
<td>The TeleTOP expert should take pro-active efforts in generating support and stimulating software use in the school.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SP_1</td>
<td>TeleTOP is an educational tool suitable for project-based education.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SS_F_1</td>
<td>TeleTOP should adequately support uploading and storing of large files in the Workspace. There should be enough disk space to store the project files.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_BS_3</td>
<td>Stop using Workspace function for uploading projects and project files; Use the Workspace for submitting interim project files but not the final product.</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_F_2</td>
<td>[ProdDesc] The workspace function of TeleTOP is improved. It can now allow submission of large files in the Workspace, i.e. 1 MB or more.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_BP_1</td>
<td>Students should submit their assignments on time or ahead of time. The continued availability of TeleTOP blurs the official notions of time. Students submit their assignments up to the last minute of the day; educational processes continue 24/7.</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_BS_4</td>
<td>Students should submit their assignments by hand. If agreed to be submitted online, it should be uploaded in TeleTOP at certain time period, i.e. 17.30</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_BP_2</td>
<td>Students should not be able to copy each other’s work or look into other group’s solution.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_BP_3</td>
<td>The Roster is not filled up completely and consistently by teachers.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_F_3</td>
<td>Functions that enable storage, uploading and downloading of files should be properly differentiated from each other. These functions are archive, presentations, roster and publications.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_BS_5</td>
<td>Students would like the teachers to give proper instructions on where to download class materials and lectures. If possible, the use of the different functions of TeleTOP that offer uploading and downloading possibilities should be consistent.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_SP_2</td>
<td>TeleTOP is an asymmetric and exclusive communication tool between teachers and students.</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_SP_3</td>
<td>TeleTOP is a single course management and material repository system.</td>
</tr>
<tr>
<td>15</td>
<td>EARLY_SS_F_4</td>
<td>Users want a seamless integration of the files and objects stored in TeleTOP with the application</td>
</tr>
<tr>
<td>16</td>
<td>EARLY_SS_F_5</td>
<td>The files in the workspace should have a structure and hierarchy so that files can be searched systematically, i.e. per group, per topic, etc.</td>
</tr>
<tr>
<td>17</td>
<td>EARLY_SS_F_6</td>
<td>It should be possible to fill-up the roster in a simpler way than the way it is: teachers have to fill in the schedule of the sessions line by line.</td>
</tr>
<tr>
<td>18</td>
<td>EARLY_SS_F_7</td>
<td>It should be possible to send an email to the whole group in one-click (see Figure 7-5).</td>
</tr>
<tr>
<td>19</td>
<td>EARLY_BS_6</td>
<td>Make use of a student assistants to aide teachers in setting up TeleTOP sites for the course</td>
</tr>
<tr>
<td>20</td>
<td>EARLY_BS_7</td>
<td>Retain the use of TeleTOP in the school despite its shortcomings.</td>
</tr>
</tbody>
</table>
EARLY_BS_1. “There should be a TeleTOP expert available for the school of Industrial Design Engineering to oversee the system deployment.” This statement embodies the pro-active strategy taken by the central body in-charge of managing the deployment of TeleTOP throughout the university. In turn EARLY_BS_2. “The TeleTOP expert should take pro-active efforts in generating support and stimulating software use in the school.” reflects the action taken by the TeleTOP expert (which is a realization of PRE_BS.1) within the school. Both requirements stem from pro-active efforts that aim to generate support for using the software, which is way of mitigating the occurrence of breakdowns.

EARLY_SP_1. “TeleTOP is an educational tool suitable for project-based education.” is the propagated image of the software serving as the content of actions taken in EARLY_BS_2. The TeleTOP implementation team through its representative in the school sought to promote TeleTOP as a solution apt for the educational processes within the school of industrial design. Largely aware of the intensive project-based courses employed by the school as its main didactical approach in teaching, the implementation team projected the concept of TeleTOP as tool suitable for project-based education through its Workspace function. This function allows students to have central space for sharing files with each other. It was thought that by doing this, it would increase the appeal of TeleTOP to teachers so that they will make use of it.

However, immediately after put into use, users encountered a major breakdown – the Workspace (EARLY_SS_F.1) was not functioning as expected. Students encountered a slow down in system performance whenever they try to upload or download large files. In some occasions, the process of uploading or downloading is terminated prior to the completion of the task. According to one of the teachers, the TeleTOP team did not take into further consideration that students involved in project-based courses produce interim outputs of 20-40 MB files. Files as large as these are apparently not supported by TeleTOP. In turn, the demand to improve the Workspace function in TeleTOP is articulated in the requirement EARLY_SS_F.1.

As a result of this breakdown, some teachers decided to stop using the Workspace for project-based work, although they continued in using other system functions. They sought other means of storing and sharing project files (EARLY_BS.3). On the other hand, some teachers continued using Workspace in its limited capacity by limiting the uploading of files into interim project files and not the final project itself. Altogether, this requirement together with the two previous ones serves to capture the irony that mar TeleTOP implementation in this period. It was intended to appeal to industrial design teachers by being able to support project-based work; however, it was precisely this function the software fell short of.

On the part of the TeleTOP design and implementation, immediate action on the Workspace issue was taken. During this period, the Workspace function of TeleTOP was improved; disk space problems were eliminated and performance is no longer hampered by uploading and downloading large files of 10 to 20 MB (EARLY_SS_F.2). While it was not clear whether this enhancement was communicated properly throughout the entire department, some teachers completely abandoned using the Workspace. On the other hand, those who continued using it did not encounter anymore problems with it – performance wise or usability-wise, unbeknownst that they are already storing large files.

The continued implementation of TeleTOP brought about new business problems. The transformation of the teaching and learning processes through TeleTOP also had negative consequences – from the viewpoint of the teachers. Because TeleTOP is always available, it blurs the notion of time and it does not enforce discipline among students, especially when it comes to deadlines. Teachers raise the issue that a number of students tend to submit their assignments to the last minute, i.e. shortly before midnight. What the teachers would like to happen is that even in an educational context in which TeleTOP is available, students should continue to submit their assignments or coursework at reasonable time in relation to the deadline (EARLY_BP.1).
Naturally, several teachers took the resolution of issues like this into the classroom (EARLY_BS_3). Some adopt the traditional way of submitting assignments – by hand. Otherwise, if the submission has to be via TeleTOP, then an agreement is made in class as to when is the real deadline of an assignment, i.e. if the deadline is Friday, then it should be in TeleTOP by 17.30 on Friday. Otherwise, the submission will be late or not accepted.

The use of Workspace also brought about new business problems. The statement EARLY_BP_2. “Students should not be able to copy each other's work or look into other group’s solution.” is triggered by a breakdown situation due to the absence of access restrictions to files in the Workspace. All students who have access to the Workspace of the course have equal access to the files that were stored in it. Therefore, groups can see through the work groups including their solutions. This is not a desirable situation for the teachers as well as the groups whose have submitted earlier. Apparently, access restriction was not a felt need at the beginning of the implementation. Upon the encountering the problem of cheating, they immediately voice out this problem resulting from using the Workspace.

As the acquaintance with the software grows, it has also become more pronounced to the regular users how TeleTOP is being used differently and inconsistently, in particular, its Roster function. Students have been accustomed to using TeleTOP and its Roster function that they find it unusual, if not sub-standard that some teachers do fill it up. Not only students notice this irregularity in use; teachers notice it too. They think that all it is highly desirable to have the Roster completely filled up. In turn, the requirement EARLY_BP_3 indicate this concern voiced out both by teachers and students about the Roster.

The statement EARLY_SS_F_3. “Functions that enable storage, uploading and downloading of files should be properly differentiated from each other. These functions are archive, presentations, roster and publications.” illustrates a usability breakdown encountered by users stemming from the lack of a sharp distinction between functions that perform or render a similar namely storage of files. The functions represented by the icons Presentation, Archive, Publication and the Roster all offer the facility of storing educational materials and making these available for download. While it may not be that confusing for the Roster because students and teachers alike regularly access it, making it the centrepiece function for TeleTOP, for the other mentioned functions it is not rather obvious for users for which specific purposes should they be used.

In turn, given that at this moment this distinction between the functions is not yet made, students put forward the suggestion that teachers should provide extra instructions on where to download materials when it is not available through the Roster. It is also the case that sometimes additional materials are used in class, i.e. software and other utilities and students don’t know specifically where to search for them. This way, the resulting requirement EARLY_BS_5. “Students would like the teachers to give proper instructions on where to download class materials and lectures. If possible, the use of the different functions of TeleTOP that offer uploading and downloading possibilities should be consistent.” is a workaround suggestion to compensate for the usability issues brought about by the software.

What has also become apparent at this stage of the implementation is the increasing use of the TeleTOP for communication purposes. Teachers make special use of the software to communicate up-to-date information with their students outside the classroom. However, this is a unidirectional type of communication – from teachers to students, and not the other way around. This particular behavior reinforces the concept of an asymmetric and exclusive communications tool (EARLY_SP_1), which is consistent with the functional affordances of the tool in relation its product specifications.

Consequently, by also carrying out administrative and non-teaching tasks through TeleTOP and by largely utilizing it to store teaching materials, teachers give rise to the concept of a course management and repository system (EARLY_SP_2). This serves as an additional software
product concept for TeleTOP consistent with its product specifications. Such product concept on the other hand is a departure from the original concept of a learning tool, i.e. an environment in which students can learn (see the discussion in Section 7.3 about the pre-implementation phase). TeleTOP was meant to be a learning tool. However, in terms of actual implementation, it is more of a course management tool. At the same time, the manner in which teachers operate the software also indicate that the course management capability is centered on a single course and not on a portfolio of courses. When teachers try to set up and maintain their courses in TeleTOP, they do it on a course per course basis. It is not possible to manage all courses at the same time in TeleTOP. In the light of this context, the formulation of this requirement is attributable in the first instance as an initiative, and as a breakdown in the next.

Also due to a usability breakdown, the requirement EARLY_SS_F_4. “Users want a seamless integration of the files and objects stored in TeleTOP with the application.”, is about having the ability to execute the files from the Workspace as they are clicked. For example, teachers would like to have the capability of automatically executing the associated software application to the file being downloaded, i.e. a CAD drawing. The current method of opening files from the Workspace in TeleTOP is to first download it to the local computer and then run the associated application. This is relatively tedious compared to the way in which most users encounter the integrated execution of Microsoft-based files, i.e. Word, Powerpoint etc when clicked from a browser.

Adding to this list of breakdowns resulting from interacting with TeleTOP is the lack of structure among the files inside the workspace which makes it difficult to search for files. As a result of this breakdown, users would like to see some form of visual organization within the files in the Workspace so that they could search for files in a more systematic manner (EARLY_SS_F_5), i.e. alphabetical or topic. It should also be possible through this visual organization to know from which group a file belongs or which project it is a part of.

The statement EARLY_SS_F_6. “It should be possible to fill-up the roster in a simpler way than the way it is: teachers have to fill in the schedule of the sessions line by line.”, is also a breakdown-induced requirement that reflects the action-intensive process of filling up the Roster. Teachers had to fill-in the Roster on a line by line basis, in which they themselves have to create the rows for each line. This process of working with TeleTOP is not consistent with the actual process creating course timetables. Usually teachers already know beforehand how many sessions they will teach and therefore the number of rows to be created in the Roster can be determined and set-up. However, it was not possible to set-up a table with a fixed number of rows in one click. The rows have to be added manually.

While TeleTOP facilitates the sending of e-mails to individual students, it was not however possible for teachers to send an email to the entire class. It was also possible to send emails to groups, provided that the teacher has a created an email group. In order to send an email to the entire class, the teacher has to select the entire list and paste them on the email software. This process becomes messy because it is not only the email address that gets copied but also the additional information, i.e. name, student number (see Figure 7-5). In turn, these extra texts need to be manually edited from the list that is pasted into the email program because they cannot be processed and the email will not be sent. In which way possible, teachers would like to have the capability of sending emails to the entire class in one-click (EARLY_SS_F_7).

The use of students assistants as specified in EARLY_BS_6 was one of the many pro-active efforts taken by the TeleTOP implementation team aimed at mitigating future troubles that might arise as a result of using the system. It also intended to reduce the burden on the part of the teachers whom by the implementation of TeleTOP ended up doing more work for their courses in comparison to situation without TeleTOP.

Finally, the remaining requirement in the list is a business solution that states the conscious
decision of the educational council to continue using the TeleTOP despite its drawbacks and the negative experiences they encountered with it (EARLY_BS_7). The context of this requirement is initiative driven because it is breakdown-tolerant situation which does not call for the removal of the source of breakdown.

![Image](image.png)

Figure 7-5. TeleTOP. List of emails page

7.3.2 Remarks

In comparison to pre-implementation, in this phase, there is also also a marked increase in the number of requirements statements from other domains aside from the software solution specification domain. It is also a period in which we have observed actual software change taking place. For example, a breakdown in the storage capacity of a software function, i.e. Shared workspace (EARLY_SS_F_1), was resolved immediately by adjusting the software (EARLY_SS_F_2). In addition, with the added knowledge about issues, we can add this additional information in describing the context surrounding each requirements statement.

7.4 Requirements at Post-Deployment: from a glider to a Boeing 747

The user base of TeleTOP at the department continued to increase as new students enter the program. Freshman students gain exposure for the first time with TeleTOP and old students remain on the system. One important characteristic of this phase is the dominance of ICT efforts taking place. For example, the student portal which is accessible via the university website is implemented. Other software applications such as the online course catalog, the official course registration system and examination registration are now made web-accessible. Through their personalized portal, students gain access to all these services.

7.4.1 Requirements statements

Various breakdown situations as well as initiatives situations characterize the formulation of requirements statements in this phase. These are given Table 7-6.

As number of students continues to increase and with teachers carrying on with the use of TeleTOP as well as the other software applications, they increasingly become more aware of the issues they are faced with their tasks. For example, teachers have expressed the need to know the official number of students enrolled in the course at the beginning of the semester. This is important for teachers to know so that they can be distributed among themselves for supervision. This problem articulation is expressed in the statement POST_BP_1: “Teachers would like to
“know the number of students who will be taking the course so that they can be distributed beforehand.”

**Table 7-6. Case C. Requirements at Post Deployment**

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POST_BP_1</td>
<td>Teachers would like to know the number of students who will be taking the course so that they can be distributed beforehand.</td>
</tr>
<tr>
<td>2</td>
<td>POST_BP_2</td>
<td>The availability of software applications that provide overlapping services create a sense of discontinuity in the process. Users want an integrated environment that supports the process or they want to deal with only just one application.</td>
</tr>
<tr>
<td>3</td>
<td>POST_SP_1</td>
<td>TeleTOP should evolve into an integrated learning suite incorporating multiple systems or their functionalities.</td>
</tr>
<tr>
<td>4</td>
<td>POST_SP_2</td>
<td>TeleTOP is an exclusive application for teachers and students only</td>
</tr>
<tr>
<td>5</td>
<td>POST_SS_Q_1</td>
<td>There should only be one interface between TeleTOP, VIST, TOST, TAST and student portal.</td>
</tr>
<tr>
<td>6</td>
<td>POST BS_1</td>
<td>Include the bureau of educational affairs in the re-design of TeleTOP and in managing the implementation.</td>
</tr>
<tr>
<td>7</td>
<td>POST BS_2</td>
<td>There should be an official or formal introduction about the different software applications used and their abbreviations explained, i.e. TOST, TAST, VIST, TeleTOP.</td>
</tr>
<tr>
<td>8</td>
<td>POST BS_3</td>
<td>Teachers maintain their own list and administration aside from the TeleTOP and VIST.</td>
</tr>
<tr>
<td>9</td>
<td>POST SS_F_1</td>
<td>Teachers should be able to create their own backup of Workspace files of their courses in DVD format.</td>
</tr>
<tr>
<td>10</td>
<td>POST SS_F_2</td>
<td>It should be possible for courses in TeleTOP to be accessible on a read-only basis to unauthenticated users.</td>
</tr>
<tr>
<td>11</td>
<td>POST SS_F_3</td>
<td>TeleTOP should provide support for mathematical symbols.</td>
</tr>
<tr>
<td>12</td>
<td>POST SS_F_4</td>
<td>TeleTOP should display most recent items, i.e. news items, newly uploaded files</td>
</tr>
<tr>
<td>13</td>
<td>POST SS_Q_2</td>
<td>There should be online help available.</td>
</tr>
<tr>
<td>14</td>
<td>POST BS_4</td>
<td>Utilize TeleTOP functions such as Poll in the classroom in order to determine students' understanding of the learning material immediately.</td>
</tr>
<tr>
<td>15</td>
<td>POST SP_3</td>
<td>TeleTOP should make itself more amenable to wireless capabilities and support mobile education.</td>
</tr>
<tr>
<td>16</td>
<td>POST SP_4</td>
<td>In the light of the integration among the three technical universities, a new digital, integrated earning and content management system will replace TeleTOP.</td>
</tr>
</tbody>
</table>

**POST_BP_2.** “The availability of software applications that provide overlapping services create a sense of discontinuity in the process. Users want an integrated environment that supports the process or they want to deal with only just one application.” It has also come to a point that while users have gotten the hang of using TeleTOP and the other applications, the availability of multiple support tools whose functions overlap each other create confusion and a sense of discontinuity in the process. For example, VIST and TeleTOP both contain information about the course—description, teacher, requirements and textbook. TOST is the application for registering for examinations and TAST is the grades register. On the other hand, as students experience it, most of these services are available and are coursed through TeleTOP: course information, reminders to register for examinations and interim grades are sometimes made available. Instead of this separation of different applications for somewhat similar and different purposes, users express the interest in having only one application.

**POST_SP_1.** “TeleTOP should evolve into an integrated learning suite incorporating multiple systems or their functionalities.” The interest in having only one application as an integrated digital environment is articulated in this statement. It is a way of resolving the earlier statement POST_BP_2. Correspondingly, one possible implementation of this software product concept is **POST_SS_Q_1.** “There should only be one interface between TeleTOP, VIST, TOST, TAST and..."
student portal.”

POST_BS_1. “Include the bureau of educational affairs in the re-design of TeleTOP and in managing the implementation.” As the multiplicity of applications related to TeleTOP has become a known issue at this stage, several users attribute this to the non-inclusion of one important stakeholder group – the bureau of educational affairs (BOZ). The original design of TeleTOP proceeds from a mental model of an educational process that incorporates only teachers and students which is reinforced in this case as a software product concept POST_SP_2. In order to correct this impression and to make TeleTOP more apt to the context, users have suggested the inclusion of the bureau of educational affairs (BOZ) in the future design of TeleTOP and in its implementation. It is hoped that by doing this, other issues such as making administrative tasks through TeleTOP, i.e. registration, grades become official and therefore centralizes the role of TeleTOP in educational process in the school.

POST_BS_2. “There should be an official or formal introduction about the different software applications used and their abbreviations explained, i.e. TOST, TAST, VIST, TeleTOP.” To some users, not only the multiplicity of applications is confusing but also the abbreviated references to them. It is common practice in the school and in the university as well to refer to the applications in their short form, i.e. TAST which is short for Tentamen Aanmeld Systeem Twente. This is a system for registering for examinations. For new students, quickly referring to these applications without a prior explanation of their purpose and what the shortened references stand for create ambiguity. It is therefore proposed that an official introduction into these systems should be given to new students or when the system is first mentioned in the classroom.

POST_BS_3. “Teachers maintain their own list and administration aside from the TeleTOP and VIST.” Resulting from a more prolonged exposure to TeleTOP An example of user adaptive behavior resulting from this implementation is given in this statement. While TeleTOP as well as TAST offer student data administration such that it is possible to maintain them online, teachers continue to retain their own personal list. Especially after the fire incident (discussed in the previous period) when most of the web and network-based applications went offline, it was deemed to be a wise idea to have maintain one’s own administration personally.

POST_SS_F_1. “Teachers should be able to create their own backup of Workspace files of their courses in DVD format.” With students’ projects files centrally stored in the Workspace, teachers saw the opportunity to have a personal backup of files from the courses. This is an area of functional improvement for TeleTOP. Specifically, one teacher put forward the suggestion of having the capability to backup project files locally or through a DVD. This is because from time to time, teachers would like to be able to refer to projects done by students in the previous years in a quick and hassle-free manner, i.e. no need for internet connection, TeleTOP login as well as searching for the files.

Additional software solution specifications are also gathered in this period. These include having the capability for unregistered users (users without login) to be able to read through course descriptions (POST_SS_F_2), support for the display of mathematical symbols (POST_SS_F_3), the presentation of newest or latest items in top of the list whenever applicable as well as the availability of online help (POST_SS_F_4). Up to this phase in the implementation, help is only available in text in the form of a reference manual. Therefore users articulate having an online help (POST_SS_Q_2).

POST_BS_4. “Utilize TeleTOP functions such as Poll in the classroom in order to determine students’ understanding of the learning material immediately.” Technological progress in terms of wireless technology is also becoming popular in use. Teachers mention that they do make use of wireless internet in the classroom and they can access TeleTOP while in the middle of the class. Usually, TeleTOP access is done outside the classroom. However, some teachers do mention that they use the Poll function of TeleTOP to test students’ knowledge of a certain concept and if
unsatisfactory, further explanation and in-depth attention will be given to the poorly understood subject matter. Access to TeleTOP in order to make use of this service is done through wireless connection.

POST_SP_3. “TeleTOP should make itself more amenable to wireless capabilities and support mobile education.” Due to the increasing popularity of wireless access to the internet, teachers have become keen on thinking along the lines of wireless possibilities for TeleTOP. In very vague terms, they think of upcoming trends, i.e. wireless technology, mobile education and form the expectation that TeleTOP should evolve into these technologies. Likewise, this outlook that anticipated upcoming trends and changes prevail such as the impending integration with the other two technical universities. Already, this forthcoming change had stirred talks and thinking about what it means to TeleTOP as far as its on-going implementation is concerned. Replacing TeleTOP with another software seems to be the general line of thinking with respect to this future development (POST_SP_4). Stakeholders foresee a new form of electronic learning and teaching environment that will soon be implemented once the fusion becomes operational.

7.4.2 Remarks

Requirements statements gathered in this case were consistent with the observation from the previous cases that requirements become more apparent when the system gets used in the organization. Most of the requirements that were gathered were from the early implementation phase, followed by post-deployment and pre-implementation. This is excluding the software product specifications listed in pre-implementation (Tables 7-3 and 7-4). Those specifications can be considered as given and they do not form as demands that have software implications.

In recovering the requirements statements, we have observed the following features and dynamics in place. First, the software was implemented from a top-down level, with the mandate coming from the larger organization; despite the autonomy the implementing unit has, it maintained the policy of the university. Second, even though the application did not meet the expectations of the users and there were problems using it, the department chose to continue implementing the software. This decision was an active choice in the sense that it took action and effort in supporting the implementation and in encouraging users to make use of the software whenever they can. Third, the prevailing concept of a digital environment is composed of multiple applications. All throughout the phases, there were recurring requirements statements that refer to software applications other than TeleTOP. Users perceive an environment composed of multiple systems in which the groupware application being studied in this case is but a part of.

The recovery of the requirements also reveal that the combined top-level mandate motivating the implementation as well as the readiness and willingness of the host unit to implement and use the software. Prior familiarity to the software also served as a precedent in enabling the access to the software, eventually reducing the risks of learning completely new software. In addition, the high cognitive capacity of users, i.e. university teachers and students, who have affinity and experience using software systems also have contributed to the sustained use of the TeleTOP. More importantly, the post-deployment support provided by the university in terms of help-desk and deployment of experts in the department contributed significantly in implementing TeleTOP. Especially when the software has a lot issues, the availability of offline helps minimize the negative perceptions about the system.

7.5 Requirements evolution: within-domain evolution

Substantial evolution is observed in this case in terms of change in most of the requirements domains. Sections 7.5.1 to 7.5.4 provide a compilation of the requirements statements per domain.
7.5.1 Business problem domain evolution

The evolution of the business problem domain in this case is characterized by a shift from a non-problem situation at pre-implementation to a problem-filled domain in the later phases of implementation (Table 7-6). At pre-implementation, there is no particular business problem that motivates the implementation of the TeleTOP. On the other hand, business problems arise and multiply once the software has been implemented. In turn, the business problem statements are most prominent prominent during early implementation, i.e. when the system is newly put into use. In most instances, the business problems we have recovered consist of mainly of statements that reflect desirable attributes of a situation. Some of these desirable attributes have direct implications on software functions and quality properties. While some are independent of the software and do not have any implications on the software but rather are expectations of what the user should be or should do.

Table 7-7. Case C: Business problem statements compilation - all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_BP_1</td>
<td>Students should submit their assignments on time or ahead of time. The continued availability of TeleTOP blurs the official notions of time. Students submit their assignments up to the last minute of the day; educational processes continue 24/7.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_BP_2</td>
<td>Students should not be able to copy each other’s work or look into other group’s solution.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_BP_3</td>
<td>The Roster is not filled up completely and consistently by teachers.</td>
</tr>
<tr>
<td>4</td>
<td>POST_BP_1</td>
<td>Teachers would like to know the number of students who will be taking the course so that they can be distributed beforehand.</td>
</tr>
<tr>
<td>5</td>
<td>POST_BP_2</td>
<td>The availability of software applications that provide overlapping services create a sense of discontinuity in the process. Users want an integrated environment that supports the process or they want to deal with only just one application.</td>
</tr>
</tbody>
</table>

7.5.2 Business solution domain evolution

The evolution of the business solution domain can be inferred from the statements compiled in Table 7-7. The compilation shows that TeleTOP implementation is motivated by business solutions. From the compilation, we can observe that the statements have evolved from strategic top-level decisions to operational measures and individual adaptation.

7.5.3 Software product concept domain evolution

Table 7-7 lists the software product concept domain requirements statements from all phases in. We can observe from this compilation that software product concept continued to change over time, with a remarkable increase in the number of ideas formed regarding the instrumental uses of TeleTOP as it gets rolled out in the organization. This observation confirms the earlier findings that prolonged interaction bring about increased knowledge about the system. With this comes the discovery and eventual formulation of new purposes and other uses for the application not originally thought of.

The increase in the number of software product concepts assigned to the system over time is indicative of a couple things about TeleTOP. First, it shows that the system has a flexible design that can accommodate other purposes. We can already see this from the very beginning when the conception of TeleTOP as a learning tool into something suitable for project-based education. Second, it is also indication that the system has limitations and users adjust their perceptions of the system, finding other purposes suitable for it. This is reflected in the turnaround in the use of TeleTOP for project-based education and focusing instead on the communication and administrative utility which are known to work.
We can observe from the evolution of software product concept that the concept is also influenced by external factors. Users relate the software with upcoming organizational change or technological trends Users relate their imminent needs with the software which they automatically perceive as a solution.

**Table 7-8. Case C. Business solution statements compilation - all phases**

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BS_1</td>
<td>TeleTOP is the mandated default learning environment in the university; all faculties and schools should make use of it.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BS_2</td>
<td>TeleTOP is already an existing resource for supporting educational tasks in the university and faculty. The school of industrial design engineering must make use of it.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_BS_1</td>
<td>There should be a TeleTOP expert available for the school of Industrial Design Engineering to oversee the system deployment</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_BS_2</td>
<td>The TeleTOP expert should take pro-active efforts in generating support and stimulating software use in the school.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_BS_3</td>
<td>Stop using Workspace function for uploading projects and project files; Use the Workspace for submitting interim project files but not the final product.</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_BS_4</td>
<td>Students should submit their assignments by hand. If agreed to be submitted online, it should be uploaded in TeleTOP at certain time period, i.e. 17.30</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_BS_5</td>
<td>Students would like the teachers to give proper instructions on where to download class materials and lectures. If possible, the use of the different functions of TeleTOP that offer uploading and downloading possibilities should be consistent.</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_BS_6</td>
<td>Make use of a student assistants to aide teachers in setting up TeleTOP sites for the course.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_BS_7</td>
<td>Retain the use of TeleTOP in the school despite its shortcomings.</td>
</tr>
<tr>
<td>10</td>
<td>POST_BS_1</td>
<td>Include the bureau of educational affairs in the re-design of TeleTOP and in managing the implementation.</td>
</tr>
<tr>
<td>11</td>
<td>POST_BS_2</td>
<td>There should be an official or formal introduction about the different software applications used and their abbreviations explained, i.e. TOST, TAST, VIST, TeleTOP.</td>
</tr>
<tr>
<td>12</td>
<td>POST_BS_3</td>
<td>Teachers maintain their own list and administration aside from the TeleTOP and VIST.</td>
</tr>
<tr>
<td>13</td>
<td>POST_BS_4</td>
<td>Utilize TeleTOP functions such as Poll in the classroom in order to determine students’ understanding of the learning material immediately.</td>
</tr>
</tbody>
</table>

**Table 7-9. Case C. Software product concept statements compilation - all phases**

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SP_1</td>
<td>TeleTOP is the tele-learning tool: it provides support in the teaching and learning process.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SP_1</td>
<td>TeleTOP is an educational tool suitable for project-based education.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SP_2</td>
<td>TeleTOP is an asymmetric and exclusive communication tool between teachers and students.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SP_3</td>
<td>TeleTOP is a single course management and material repository system.</td>
</tr>
<tr>
<td>5</td>
<td>POST_SP_1</td>
<td>TeleTOP should evolve into an integrated learning suite incorporating multiple systems or their functionalities.</td>
</tr>
<tr>
<td>6</td>
<td>POST_SP_2</td>
<td>TeleTOP is an exclusive application for teachers and students only</td>
</tr>
<tr>
<td>7</td>
<td>POST_SP_3</td>
<td>TeleTOP should make itself more amenable to wireless capabilities and support mobile education.</td>
</tr>
<tr>
<td>8</td>
<td>POST_SP_4</td>
<td>In the light of the integration among the three technical universities, a new digital, integrated earning and content management system will replace TeleTOP.</td>
</tr>
</tbody>
</table>
7.5.4 Software solution specification domain evolution

Table 7-8 is a compilation of software solution specification statements consisting of both product specifications which already exist and those that are still desired. For purposes of simplification, we only list the main statements of the pre-implementation functional product specifications given in Table 7-3 in this compilation.

We can gather from this compilation that the software solution specification domain has evolved from product specifications to new functional and quality specifications over time. To begin with, the implementation started with no new functional or quality specifications; instead, it began with a pre-defined set of product specifications because TeleTOP is already a functioning application. The apparent mindset is that whatever the current set of functionalities TeleTOP has will just be carried over and used.

One of the remarkable features of evolution captured in this domain is the process of software update. At early implementation, we have observed software change taking place as a result of requirements change. The software was changed (EARLY_SS_F_2) in order to address the demand that the Workspace feature of the system should allow for large files to be stored (expressed in EARLY_SS_F_1). In other words, the product description has also evolved. This evolution is a reaction to the mismatch between the promised functionality of the software and its actual performance.

As in the previous case studies, new software solution specifications arise after the application is put into operational use in this case. During the early implementation and post-deployment phases, new functional and quality specifications arise. In most instances, these new requirements results from task breakdowns in wherein users were not able to perform their desired operations with system as well as achieve the desired outcomes of these operations. What is remarkable about these system operations is that they extend beyond the software of interest. The articulation of these new specifications is influenced by the users’ experience with using other software applications in parallel. In return, these new specifications highlight the key feature of this case, namely the implementation of groupware in this case forms part of an integrated environment. It is much more useful to refer to an electronic environment that is composed of multiple applications that altogether provides holistic support for the educational processes in the Institute. Likewise, the specifications reflect also the choppy measures taken to support the entire educational processes and activities that almost every teacher or student undergoes. There are separate applications for almost every core activity in the process, i.e. testing, grading and course administration. Users in the end encounter breakdowns when performing one activity to the other because they must not only switch applications but also remember the appropriate applications for the task.

7.5.5 Remarks

The compilation of requirements statements and the analysis of within-domain evolution in this case study give the impression that all requirements domain change over time. The analysis provides two interesting observations which reflect the particular properties of the implementation in this case. Firstly, it is an implementation with no specific requirements to begin with but with software product specifications already known and given. Therefore, we have observed an evolution from no requirements, i.e. business problem domain, to many requirements over time. Secondly, this case also provided us with an example of software update taking place captured as an update in the evolution of software solution specification domain.
Table 7-10. Case C: Software solution specification statements compilation - all phases

<table>
<thead>
<tr>
<th>Item</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_SS_F_1</td>
<td>[ProdDesc] TeleTOP is a role-based web application; it recognizes two types of roles:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>teachers and students.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_SS_F_2</td>
<td>[ProdDesc] Course information, registration and administration</td>
</tr>
<tr>
<td>3</td>
<td>PRE_SS_F_3</td>
<td>[ProdDesc] Course planning, logistics and study support</td>
</tr>
<tr>
<td>4</td>
<td>PRE_SS_F_4</td>
<td>[ProdDesc] Off classroom communication</td>
</tr>
<tr>
<td>5</td>
<td>PRE_SS_F_5</td>
<td>[ProdDesc] Interactive online communication</td>
</tr>
<tr>
<td>6</td>
<td>PRE_SS_F_6</td>
<td>[ProdDesc] Collaboration</td>
</tr>
<tr>
<td>7</td>
<td>PRE_SS_Q_1</td>
<td>[ProdDesc] TeleTOP is available in both English and Dutch versions.</td>
</tr>
<tr>
<td>8</td>
<td>PRE_SS_Q_2</td>
<td>[ProdDesc] TeleTOP has a text-based manual available.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_SS_F_1</td>
<td>TeleTOP should adequately support uploading and storing of large files in the Workspace.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There should be enough disk space to store the project files.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_SS_F_2</td>
<td>[ProdDesc] The workspace function of TeleTOP is improved. It can now allow submission</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of large files in the Workspace, i.e. 1 MB or more.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_F_3</td>
<td>Functions that enable storage, uploading and downloading of files should be properly</td>
</tr>
<tr>
<td></td>
<td></td>
<td>differentiated from each other. These functions are archive, presentations, roster and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>publications.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_SS_F_4</td>
<td>Users want a seamless integration of the files and objects stored in TeleTOP with the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>application.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_SS_F_5</td>
<td>The files in the workspace should have a structure and hierarchy so that files can be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>searched systematically, i.e. per group, per topic, etc.</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_SS_F_6</td>
<td>It should be possible to fill-up the roster in a simpler way than the way it is: teachers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>have to fill in the schedule of the sessions line by line.</td>
</tr>
<tr>
<td>15</td>
<td>EARLY_SS_F_7</td>
<td>It should be possible to send an email to the whole group in one-click. (See Figure 7-5).</td>
</tr>
<tr>
<td>16</td>
<td>POST_SS_Q_1</td>
<td>There should only be one interface between TeleTOP, VIST, TOST, TAST and student portal.</td>
</tr>
<tr>
<td>17</td>
<td>POST_SS_F_1</td>
<td>Teachers should be able to create their own backup of Workspace files of their courses in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DVD format.</td>
</tr>
<tr>
<td>18</td>
<td>POST_SS_F_2</td>
<td>It should be possible for courses in TeleTOP to be accessible on a read-only basis to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unauthenticated users.</td>
</tr>
<tr>
<td>19</td>
<td>POST_SS_F_3</td>
<td>TeleTOP should provide support for mathematical symbols.</td>
</tr>
<tr>
<td>20</td>
<td>POST_SS_F_4</td>
<td>TeleTOP should display most recent items, i.e. news items, newly uploaded files</td>
</tr>
<tr>
<td>21</td>
<td>POST_SS_Q_2</td>
<td>There should be online help available.</td>
</tr>
</tbody>
</table>

7.6 Impact relations

The mechanisms of requirements evolution as impact relations observed in this case are compiled in Appendix C to I and are discussed as follows (Sections 7.6.1. t through 7.6.7).

7.6.1 M1. Business problem resolved by business solution

We found 2 instances of business problems resolved by business solutions in this case (Appendix C: Case C). The first problem instance is a side-effect of TeleTOP use: the blurring of the official notions of time (EARLY_BP_1). This is an undesired effect for the teachers but a favorable one for students. However, since the process owner or the determining authority for this task is the teacher, some of them took the initiative of drawing a classroom policy that requires students to submit assignments by hand with a specific deadline (EARLY_BS_4. “Students should submit their assignments by hand. If agreed to be submitted online, it should be uploaded in TeleTOP at
a certain time period, i.e. 17.30.”). The second instance is the incomplete and inconsistent filling up by teachers of the Roster (course timetable) in TeleTOP (EARLY_BP_3). This is resolved by using student assistants to assist teachers with course administration including filling in the Roster (EARLY_BS_6). These examples are observed to be breakdown generated.

7.6.2 M2. Business solution motivating a software product concept

Various instances of initiative-driven business solution statements have been extended by several software product concepts (Appendix D: Case C) have been observed in this case. For example, promoting the use of TeleTOP (EARLY_BS_2) implies finding appropriate applications for the groupware, specifically one that fits the process of the department. In this case, the strategy is to align the software to the project-based approach in design engineering education (EARLY_SP_1). The other instances of this impact relation are about the consequence of a decision to continue using the software (EARLY_BS_7) and then in turn think of ways on how the software can be re-packaged as product for other educational purposes (POST_SP_4 and POST_SP_4).

7.6.3 M3. Software product concept realized by a chosen software solution specification

The realization of a certain software product concept by specifications of software properties is a requirements change mechanism that is rather evident in this case (Appendix E: Case C). For example at pre-implementation, the association between the software product concept (PRE_SP_1) and the software solution specification domain t is quite apparent. The product properties of TeleTOP as 'given' software that already being used in other departments in the university (Table 7-3 and 7-4; Appendix E: Case C, Item 20) are more or less implementations of what is understood to be a tele-learning tool for supporting teaching and learning processes (PRE_SP_1). Similarly, this product concept is further worked out in additional software solution specifications in later phases, i.e. EARLY_SS_F_1 (Appendix E: Case C, Items 21 and 22). It is interesting that note that as the software product concept changes in later phases, additional specifications were also being formulated and at some point, get implemented in order to bring about the product idea into fruition. For example: EARLY_SP_1. “TeleTOP is an educational tool suitable for project-based education” is realized by EARLY_SS_F_2. “[ProdDesc] The workspace function of TeleTOP is improved; it can now allow submission of files in the Workspace, i.e. 1 MB or more.” (Appendix E: Case C, Item 24). These instances of M3 in this case are all initiative-driven.

7.6.4 M4. Software solution specification supported by a business solution

Even though TeleTOP is an already known software application to some users in the department, rolling out it officially as a support tool also required support and intervention. The instances of M4 given in Appendix F are examples of actions and decisions taken to stimulate the implementation of TeleTOP. They were triggered both by breakdowns and initiatives. By extending TeleTOP to the Department of Industrial Design Engineering, the university further reinforces its implementation and use as an initiative (Appendix F: Case C, Item 7). In certain instances, a decision can also mean taking practical approaches such that the disruption caused by an improper functioning system is minimized and without discrediting the system so that it can remain in continued use. To illustrate this point, we refer to the decision to stop using the Workspace function in TeleTOP when submitting final projects but rather smaller project components (EARLY_BS_3). This is in response to the breakdown encountered when uploading large files to the Workspace, which the system was not able to support because there was not enough disk space (EARLY_SS_F_1). The rest of M4 instances are examples of ways to work around the limitations of TeleTOP and looking for other opportunities in which TeleTOP can be
used. For example, the use of student assistants (EARLY_SS_F_6 ➔ POST_BS_4), asking teachers to give specific instructions on where to download lecture slides in TeleTOP (EARLY_SS_F_3 ➔ EARLY_BS_5) and experimenting with features to find out whether they can solve teaching problems (PRE_SS_F_6 ➔ POST_BS_4).

7.6.5 M5. Business solution leading to improved software solution specifications

The association between the business solution domain and the software solution specification is also established in this case in a way that the former provides useful hints in formulating statements for the latter. To cite an example from the M5 impact relations instances (Appendix G: Case C): EARLY_BS_5. “Students would like the teachers to give proper instructions on where to download class materials and lectures. If possible, the use of different functions of TeleTOP that offer uploading and downloading possibilities should be consistent.” elaborates the need for online help (POST_SS_Q_2). At the same time, business solutions verbalized in terms of practical decisions such as re-defining how a function can be use help in adapting the software to better fit the needs of users (see Appendix G: Item 16, EARLY_BS_3 ➔ EARLY_SS_F_2 on the resolution of the Workspace issue). The instances of M5 impact relations happen to be breakdown triggered.

7.6.6 M6. Software solution specification leading to new business problem(s)

This impact relation is foremostly about how implementing and using a software can lead to breakdowns. In this case, we can derive vivid examples of this impact relation at work (Appendix H: Case C, item 17). The impact relation between EARLY_SS_F_2. “[ProdDesc] The workspace function of TeleTOP is improved. It can now allow submission of large files in the Workspace, i.e. 1 MB or more.” and EARLY_BP_2, “Students should not be able to copy each other's work or look into other group's solution.” tells of the side effects of using TeleTOP in the teaching and learning process. On the other hand, it can also be an instance of an initiative turning into a breakdown (Appendix H: Case C, item 16). The statement EARLY_SS_F_2. “[ProdDesc] Course planning, logistics and study support.” is a given specification of TeleTOP whose implementation stems from an initiative. However, this functionality brings about issues in use. Therefore, in this impact relation, the trigger that be of an initiative of a breakdown. Even though it may seem that we are looking for a breakdown in this impact relation, the breakdown however is the outcome of the resolution. The impact relation is instantiated by initiative that had adverse effects, therefore a breakdown. This example highlights property of a breakdown as an imposing and dominant issue.

7.6.7 M7. Business problem resolved by (re)defining the software product concept

Defining a software product for a recognized problem is a requirements change mechanism observable in this case in the later phases of implementation (Appendix I: Case C). When the increasing number of educational applications began to irritate users because of redundancy and the inconvenience of signing in to more than one applications for one particular purpose (POST_BP_2), the mental model of an integrated environment comes to mind (POST_SP_1 and POST_SP_2). For this impact relation in this case, the motivating trigger for change is breakdown.

7.6.8 Remarks

The impact relations that convey various mechanisms requirements change have all been accounted for in this case. What we have identified in our framework as issues that trigger requirements change have been incorporated in the discussion of each impact relation. This way,
for each impact relation instance, we were able to specify the additional property of its resolution as a breakdown or an initiative. Our compilation shows that on the overall, the most of the impact relations are evenly triggered by breakdowns and initiatives. The impact relations M1: Business problem resolved by business solution, M5: Business solutions leading to improved software solution specifications, and M7: Business problem resolved by (re)defining software product concept are breakdowns. On the other hand, the impact relations M2: Business solution motivation a software product concept and M3: Software product concept realized by a software solution specification and M6: Software solution specification leading to new business problems are initiative-driven according to this case. This case study shows also that an impact relation alone can result both as a resolution of an initiative or a breakdown, i.e. M4: Software solution specification supported by business solutions.

7.7 Discussion

While this is the third case study conducted for this research, it is however the first case study in which the complete version of the conceptual framework has been applied. It is also the first time at this stage in our research that our conceptual framework remained unchanged with no major theoretical updates being introduced. The main contribution of this case is its further affirmation of the usefulness of the framework as a conceptual tool in documenting and examining requirements change processes. To conclude this chapter, there are two things we would like to further discuss. The first is a clarification of the mechanism to resolve an impact relation: how the issue prompting requirements change is identified either as breakdown or initiative. Together with this, we also would like to discuss whether impact relations can be distinctively identified with a particular issue (Section 7.7.1). Second, in consideration of research question L2Q8 (6.4.2), we would like to identify unique attributes of this case and the groupware implementation that contributed to its prolonged use (7.7.2).

7.7.1 Impact relation resolution mechanisms

In Section 7.6.8, we paid closer attention to the resolution of impact relations as breakdowns and initiatives based on the observed instances of such change mechanisms. We have found that the 7 impact relations are evenly triggered by breakdowns and initiatives. At the same time, an impact relation can also result from the resolution of both breakdowns or initiatives.

Usually, we define the issue resolved by the impact relation on the basis of the source requirement. However, it can happen that the issue raised by the updated requirement in another domain may appear more dominant than the issue raised by the source requirement. An example of this is impact relation M6. Software solution specification leading to new business problem in which the source requirements are initiative driven, but they lead to breakdowns (Section 7.6.6). Because M6 as a mechanism implies the presence of a breakdown, we assume this impact relation to be generally breakdown driven. However, several instances of M6 impact relation in this study (Appendix H) show that this is not case. In this impact relation, the trigger remains to be the issue raised by the source requirement. The property of being a breakdown or an initiative is defined by the source requirement which instantiates the impact relation in the first place.

Based on the previous clarification, we can say that whether an impact relation is a typical breakdown or initiative is based on how the instances are resolved. On the basis of this case alone, in Section 7.6.8, we have observed that some impact relations are singularly motivated by a specific issue, i.e. purely breakdowns or typical initiatives. However, in order to generate more a conclusive finding, a comparison of observations from other cases is needed.
7.7.2 Lessons learned

Several studies of course management implementations and groupware implementations in general suggest several key success factors and guidelines on how to manage software implementation (Collis 2001; Rogers 2003; DeLone & McLean 2003; Holsapple & Lee-Post 2006; Serce & Yildirim 2006; Malikowski, et al. 2006; Chang et al. 2010; Naveh, et al. 2010). We recognize some of these factors in this case, i.e. critical mass (Rogers 2003; DeLone and McLean 2003), culture and norms (Collis 1999; Malikowski et al. 2006), commitment, (Chang et al. 2010) and provision of communication and knowledge infrastructures (Holsapple & Lee-Post 2006; Serce & Yildrim 2006). In the light of these findings, we would like to identify the unique properties of the implementation in this case that contributed to the system's prolonged use. At the same time, we would like to identify several challenges the system is faced with which has implications on its design, i.e. upgrades and implementation lifecycle, i.e. eventual replacement.

Phased implementation

In the light of the entire implementation of TeleTOP in the university, the software was rolled out in small steps, starting from one faculty to the next. As mentioned in the introduction, TeleTOP was first used in the faculty that developed it. The results were propagated to the university board which promoted its implementation one faculty at a time. Therefore, its implementation to a new faculty such as the Department of Industrial Design Engineering is just an extension of an on-going implementation.

Top level endorsement support realized at operational level

While on the whole, the idea of implementing TeleTOP is an initiative to introduce innovation, its eventual university-wide implementation is not possible without the endorsement and support of the university. This is realized by formalizing the implementation of TeleTOP and making it a compulsory tool in each faculty. Likewise, the support is further realized at the operational level. Helpdesk was established by placing TeleTOP contact person and expert for each faculty to support the implementation and assist users.

Prior exposure to system use

In this implementation, the groupware was not really a new and unknown system to users. The user base at the department is a mix of old and new users. This establishes system accessibility and familiarity which makes the transition to use a lot easier.

Care and concern for the system

Even though the system was introduced originally without a thorough analysis of the unique educational processes in the department and therefore it failed to address some of these (see 7.3 Workspace issue EARLY_SS_F_1), it was nonetheless promoted to be used. According the Education Coordinator of the department, TeleTOP is an agenda item in their regular meetings. In these meetings for example, they made the conscious decision to continue using TeleTOP even though it has a lot of problems. This response can be recognized in Ciborra’s (1996) observation of human behavior in groupware implementations; that users and sponsor’s cared for the system. They just don’t simply abandon or stop using the system even though it does not completely suit their needs.

Challenges

In this implementation, there are several challenges that are brought to attention that can impact
system design and its future implementation. These are:

**Design issues**

There are several things in system design that needs to be improved. For example, the system only recognizes only two roles: teachers and students. On the other hand, the educational administration processes involve more than just teachers and students. It is interesting to mention that the department registrar is involved in the use of TeleTOP, i.e. course offering, course description, etc. but they cannot log into the system. One teacher was very critical about the system which he considers a weak product, comparing it to a hang glider being developed into a Boeing 747. Coming from an industrial engineering background, this teacher thinks that TeleTOP is not a commercial product that can be put out in the market for mass production.

**Deviation in system design intention and actual use**

Going back to the original design intention for TeleTOP, we mentioned that it was developed as an educational tool to support the learning process. It was a tool for learning. However, in its actual implementation in the department, it was used more for addressing administrative processes. It was used as a course management system. Therefore, there is a difference between the original software product concept and actual product idea for which the software is being used. This issue can also be seen as an opportunity for further development and upgrade.

**Integration and differentiation with other applications**

The issue of integration and differentiation with similar and related applications is a continuing challenge not only in course management or learning management applications. The number of specialized applications being used by organizations continue to increase. This is apparent in this case and have been articulated by users as an issue. There are separate applications for exam registrations, list of subjects per curriculum, etc., which are not linked to each other. As of this research, there are activities underway to link the applications and simplify the user interface. For such issue, a single sign-on can be considered a solution.
CHAPTER 8

CASE STUDY D: ORACLE ILEARNING @ FDS ACADEMY

Business priorities drive implementation lifecycle

Oracle iLearning is a web-based learning management system (LMS) that can be implemented by user organizations as internal in-house system or as hosted application on a subscription basis. In this fourth case in our study, we will examine the implementation of Oracle iLearning within the training department of a software development company. The case provides an interesting example of a COTS groupware implementation lifecycle ending in system abandonment. While there is a general view within the training department that a learning management system is indispensable to training and that the system has proven itself to be useful in the course of its six-year implementation, severe performance breakdowns in the near-end phase, shifting business priorities and a continuously changing organizational landscape bore upon the system’s operational feasibility and desirability.

8.1 Case background

Financial Data Systems (FDS) is a software development company that specializes in solutions for centralized management of financial data. It offers financial institutions worldwide a risk management infrastructure for handling the acquisition, transformation, storage, validation, cleansing and distribution of large volumes of stock market data. This applies to dynamic price data, e.g. share prices open, high low, close, bid and ask, as well as reference data, i.e. descriptive non-price attributes of a financial instrument such as issue name, issuer, currency, the exchange (NYSE, FTSE), etc. In a general problem scenario, financial institutions such as banks and their treasury departments would have a portfolio of financial assets whose performance in the market they monitor. They subscribe to data vendors such as Reuters (http://www.reuters.com) or Bloomberg (http://www.bloomberg.com) which provide them with regular feeds of stock market data. Usually, banks subscribe to more than one data vendor for purposes of completeness and reliability. Issues arise when it comes to data integrity and the need to compare values from multiple sources of data. To begin with, data vendor feeds are not consistent in format, i.e. EDI format, fixed fields, and in presenting attribute values, i.e. Reuters – total no. of shares, Bloomberg – shares in tens of thousands. When it comes to data quality, raw vendor data can also contain errors, i.e. misplaced decimal, additional zero. In turn, banks can make use of FDS’s data management software which incorporates a framework for storing, replicating, validating, analyzing and consolidating data. One end result, for example, is a golden copy, which is a composite record of one particular financial instrument whose attribute values and transformation
from raw data sources can be traced and verified. Therefore, the resulting golden copy can be trusted. The core product has built-in validation functions as well as room for creating custom validation functions that checks errors including rules for analyzing and cleansing data. The distinction between the content, i.e. the data in the system which the business revolves around, and the logic, i.e. the solution architecture embedded in the software is reflected in the broad set of users the system caters to. For the business side, FDS data management software users include business or financial analysts and data cleansers or operators. For the IT side, which is mainly about system customization, localization, interface development, implementation and maintenance, FDS users include IT engineers, software developers and system administrators. Consequently, these users with the inclusion of their managers make up the target group of FDS training services.

In just a few years after its incorporation, FDS was able to gain a considerable base of large financial institutions in Europe as customers. This initial success can be attributed to the unique solution approach towards financial data management which is recognized by industry awards it won. From its main operations in a rural village in Friesland, FDS expanded aggressively opening offices in Amsterdam, London and New York. In April 2007, it was bought by its largest customer (in terms of software license and privately held assets) based in the United States. This led to a series of organizational changes of which some are relevant to our case study and will be discussed accordingly.

8.1.1 Implementation setting: FDS Academy

FDS Training is the training department within Financial Data Solutions (FDS) that is responsible for the delivery of software product training for its customers, partners and employees. It is made up of a training manager (1), training coordinator (1), training developers (2) and trainers (2). One member of the team has the dual role of a training coordinator and training developer. Therefore, the teams consists of a total of five. When it was created, it was called the FDS Academy. This remained until the company buy-out in which the ensuing re-organization appended the training group to the beta-testing department, namely, Solution Center. However, FDS Academy was retained together with its original manager and became part of Human Resources (HR). The rest of the original training team became FDS Training with the Solution Center manager acting as its new manager.

8.1.2 Stakeholders

The stakeholder in this case is delineated to be the training team who is the system sponsor and at the same time system user who make a broad use of the application. On a regular basis, trainees and potential customers also make use of the system but these are non-permanent users, i.e. their accounts are deleted once the training is finished. Trainees consist of customer-sent trainees, partners and own employees. Their feedback regarding the system was communicated through the trainers, training coordinator and the evaluation form that they need to fill-in at the of each training session.

8.1.3 Groupware application description

Oracle iLearning was implemented at FDS as an externally hosted application by the service provider. Through a login and password, FDS has access to its own learning management system site which it can completely administer. As a COTS application, Oracle iLearning has a built-in learning management system (LMS) architecture that promotes eLearning by enabling system sponsors to create curricula, upload eLearning modules, create training sessions, create student accounts, develop assessments and tests items. More information about the product can be found on its website: http://ilearning.oracle.com.
8.1.4 Methods and protocols

This case study is different from the previous three cases in terms of data gathering because the researcher has been a part of the training team. In-depth observations, software artifact use, document inspection, interaction and conversations (formal and informal) with the other team members and trainees were the means used to gather data. We made use of the final conceptual model and the updated research questions to structure, analyze, and report this case.

8.2 Requirements at pre-implementation: “we want training”

The pre-implementation phase in this case dates back to the creation of the training department and the motivations for its existence. It is a period that can be characterized as “we want training”, which sums up the increasing awareness of customers of their need to learn more about the FDS products that they bought in order to productively put them into use. It is also the felt need within FDS which as an organization is expanding. New employees needed to be brought up to speed in terms of knowledge and competence about the organization and their respective functions. Ultimately, these served as precursor to the acquisition and implementation of the Oracle learning management system at FDS. The surrounding context and the motivations provide the case with relevant requirements statements listed in Table 8-1. Each of the requirements statement is discussed accordingly in the succeeding paragraphs.

Table 8-1. Case D: Requirements at Pre-implementation

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BP_1</td>
<td>Aside from consulting services, customers demand for more structured and formal knowledge transfer efforts such as training and documentation regarding the use and implementation of FDS software.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BS_1</td>
<td>To address customer demands, FDS should set-up a training and documentation department: FDS Academy.</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BP_2</td>
<td>For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.</td>
</tr>
<tr>
<td>4</td>
<td>PRE_BS_2</td>
<td>Existing resources should be used i.e. in-house technical writer to lead the creation and set-up of the training department.</td>
</tr>
<tr>
<td>5</td>
<td>PRE_BS_3</td>
<td>Consultants with deep FDS system knowledge should become trainers and the vacant room in the building can become a training lab.</td>
</tr>
<tr>
<td>6</td>
<td>PRE_BP_3</td>
<td>The new training department is not experienced with the provision of training services and development of training materials.</td>
</tr>
<tr>
<td>7</td>
<td>PRE_BS_4</td>
<td>Assistance of third party experts can be used to help develop the AC Academy training curriculum.</td>
</tr>
<tr>
<td>8</td>
<td>PRE_SP_1</td>
<td>Training will be class-room based, instructor-led training.</td>
</tr>
<tr>
<td>9</td>
<td>PRE_BS_5</td>
<td>For the training lab, there should be one PC per trainee.</td>
</tr>
<tr>
<td>10</td>
<td>PRE_BP_4</td>
<td>Training handouts should be available and accessible to trainees during and after the training.</td>
</tr>
<tr>
<td>11</td>
<td>PRE_BS_6</td>
<td>Trainees will receive the training materials in electronic format on USB stick but excluding the FDS software</td>
</tr>
<tr>
<td>12</td>
<td>PRE_BP_5</td>
<td>Customers would like to have training conducted in their own offices.</td>
</tr>
<tr>
<td>13</td>
<td>PRE_BS_7</td>
<td>As business opportunities grow, FDS creates new functions and hires more staff.</td>
</tr>
<tr>
<td>14</td>
<td>PRE_BP_6</td>
<td>FDS feels the need to bring (new) employees up to speed in term of knowledge and competency.</td>
</tr>
</tbody>
</table>

As a young company with a growing customer base, FDS was faced with the challenge of providing additional services such as training to support its main software product line. Founded in 1990, FDS is a software company that survived the .com crash and managed to succeed afterwards. The expanding customer base indicates that its software solution addresses a certain
need in the industry. On the other hand, the software is known to be complex and difficult to use. It has an open architecture that allows for customized extensions and interfaces. It also has limited documentation. One of its early customers who on purpose did not go for an IT training and years later called for training and admitted that:

We had end-user training in the past but not IT training. We have a lot of good developers. However, we tried to figure out how to create custom interfaces using Java but it was quite difficult. It was difficult for us. We would like to have training preferably onsite. – Customer A, project manager, requesting for training in 2007.

PRE_BP_1. “Aside from consulting services, customers demand for more structured and formal knowledge transfer efforts such as training and documentation regarding the use and implementation of FDS software.” This statement a breakdown-driven requirements statement on two counts. The first is the difficulty experienced by customers in putting FDS data management suite into optimal use and implementation due to lack of knowledge and insight about the system. The second is the lack of available services within FDS to immediately address the learning needs of its customers. While it deployed its consultants onsite to assist customers in implementation, this did not satisfy the growing need for more standard knowledge transfer sessions that was generally applicable to a variety of users and customers. Specifically, customers were demanding for training. On the other hand, this demand created an opportunity for FDS.

In order to address the growing demand for training by its clients, FDS decided to set-up a training department. This is documented in PRE_BS_1. “To address customer demands, FDS should set-up a training and documentation department: FDS Academy.” As a young organization, FDS had room for growth and expansion. Setting up a training team was an opportunity which according to our requirements classification is a business solution aimed at addressing the problem customer demand for training. This action led to the creation of the FDS Academy.

PRE_BS_2. “Existing resources should be used i.e. in-house technical writer to lead the creation and set-up of the training department.” FDS Academy was created when the CEO asked its technical writer to spearhead the creation of the training department. Therefore, FDS Academy in the beginning was both a training and documentation team with the said technical writer as its manager. This action is captured in the statement PRE_BS_2 which can be seen as an initiative as the CEO looked first internally within the capabilities of its own organization.

PRE_BP_2. “For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.” Setting up a new department brings along requirements with it. New competencies and resources were needed by the new training department such as trainers, training materials and a training venue. These were problem items for the new training manager. In a similar approach, some of the resource constraints were resolved internally. Before the idea of creating a training department came about, consultants deployed onsite have already been providing some of form of knowledge transfer to customers on a less structured basis. Professional services consultants sometimes provide workshops to a small group of users or developers at the customer site. Therefore, it was decided that the trainer role can be fulfilled by a consultant. Actually, in the long run, this decision turned out to be the most desirable because professional services consultants have deep knowledge about the product and of customer requirements. They are ideal for the trainer role. However, they might need additional teaching skills training and techniques to become trainers. FDS Academy was able to secure one consultant to become a full-time trainer. A few years later when this trainer became ill for a long time, FDS Academy was able to get another consultant as replacement. At the same time, there was extra space in the building which was converted into a training center. The FDS office used to be an old police building with a ground floor extension for 3 prison cells. That area is not being used and it became the training center fitted with training equipment. These two
decisions are business solutions that aim to fill the resource requirements of the new training department (PRE_BS_3). We consider this to be an initiative because it offers one of the many possible solutions to address resource requirements.

PRE_BP_3. “The new training department is not experienced with the provision of training services and development of training materials.” On other hand, some of the resources and competencies needed by the new training department cannot be sourced or handled internally. For example, the production of training materials and more importantly, the specific know-how of designing appropriate training are competencies the FDS Academy was lacking. These requirements serve as bottlenecks that impede the launch of the training department.

The FDS Academy sought the services of a London-based consulting company to help set-up its training curriculum (Table 8-1: PRE_BS_4). This resulted into the development of an end-user course containing 6 modules. Two years later, FDS Academy added an IT course consisting of another 6 modules to its training curriculum. The training materials that were developed were mostly in the form of presentation materials. The general idea was that training will be classroom-based and instructor-led (Table 8-1: PRE_SP_1). In turn, training will be held at the FDS training center.

PRE_BS_5. “For the training lab, there should be one PC per trainee.” The FDS training center was fitted with training equipment such as computers. It was also the idea that there should be one PC per trainee. At that time, the training lab had 8 training PCs which means the same amount of trainees, although there have been instances in which there are more participants than PCs. In such cases, two trainees would share one training PC. With training PCs available, trainees could follow the lectures on their training PCs. It was a requirement that trainees should have access to the handouts before and after the training (Table 8-1: PRE_BP_4). That why, in most training sessions, trainees received the training materials in physical form, i.e. paper format or CDROM. FDS Academy tried to avoid the hassle of printing handouts by providing them in USB sticks. This way, trainees received the handouts and other training materials in digital format with the exception of the FDS software which requires license (Table 8-1: PRE_BS_6).

PRE_BP_5. “Customers would like to have training conducted in their own offices.” At some point, customers also began requesting for onsite training. Usually, the customer would like to train a larger group of participants and have a trainer come to them which much more cost-effective. There are also practical considerations for this request, i.e. the trainees are physically close to the implementation and the trainer can visualize the context. Given that the trainer was also a consultant before, training can be customized to certain needs. In the years that followed, it had become a common pattern unique to FDS training, which was also a source of tension, that there is a blurred line between standard training and consultancy.

With the creation of the FDS Academy, training had become part of the products and services offering of FDS. After its creation, the demand for training continues as the customer base for software license agreements continues to expand. At some point, the CEO made the comment that the training department is the most important department in FDS. As business opportunities grew, FDS also expanded organizationally by creating new functions and hiring more staff (PRE_BS_7). It opened offices in Amsterdam and New York. Such initiative seems to be a natural course of action taken by most organizations: when business expands, the organization must also expand. In turn, the FDS Academy’s staffing also grew with the addition of a training developer and a training coordinator to its team. With the foreseen growth in sales and services and a rapidly expanding team, FDS was confronted with the need to train and equip its new employees with appropriate skills and competencies in the shortest time possible (Table 8-1: PRE_BP_6).
8.3 Requirements at early implementation: capacity building and expansion

Setting the stage for the early implementation phase was the search for solutions and approaches that would address the organizational need for capacity building as its opportunities grow. Taking off from the last requirement in the pre-implementation phase (Table 8-1: PRE_BP.6), the task of expediting the process of inducting new employees to the organization fell within the responsibility of the FDS Academy. Employees needed to be oriented about the company and prepared functionally for their tasks within FDS. In turn, the FDS Academy began considering other forms of learning, i.e. e-learning, and learning platforms such learning management systems (LMS) in order to address the organization and its customers’ needs. This resulted in the selection, choice for and implementation of Oracle iLearning as an e-learning environment. In Table 8-2, we aim to reconstruct the events and decisions in this period through the requirements we were able to gather from this case.

As discussed in the beginning of this section, FDS focused on internally building the capacity of its growing organization. In addressing the need to induct new employees in a more time-efficient manner, the FDS Academy sought solutions in terms of e-learning and got acquainted with training delivery mechanisms such as learning management systems (LMS) (Table 8-2: EARLY_SP.1). As the manager also emphasized, “We’d also like to follow the trend.” To go about these thought of solutions, the FDS Academy engaged the services of a London-based consultancy company for a market study of LMS solutions (Table 8-2: EARLY_BS.1). The FDS Academy together with this consulting company drew up a procedure for selection, made an inventory of requirements and issued a request for proposal (RFP). We were able to recover the RFP and the list of LMS requirements are listed in Table 8-3.

The RFP contained software solution specifications that more or less capture what FDS Academy wants of learning solution such as an LMS in relation its specific needs. We will not go into each of the specifications but we will discuss a few. A very important requirement for example is the preference for a hosted environment because it does not want to tax its internal IT infrastructures (Table 8-3: EARLY_SS_Q.3). Next to this is seamless integration with various e-learning authoring tools as FDS Academy intends to develop dynamic, media-rich and complete instructional modules (Table 8-3: EARLY_SS_Q.4).

The rest of specifications in the RFP provide details of desired functionality, features and appropriate support for specific learning activities such as assessments, classroom management and reporting. While expressed in broad terms i.e. 24x7 support (Table 8-3: EARLY_SS_Q.6), doesn’t require a plugin (Table 8-3: EARLY_SS_F.21) scheduling (Table 8-3: EARLY_SS_F.20), etc, the RFP requirements specifications provide basic information that a supplier can work with.

EARLY_BP.1. “FDS partners should also get training on FDS data solutions.” Aside from training its own personnel, FDS is also faced with the problem of training its Partners. FDS entered partnerships with IT consulting companies in order to expand its capacity to deliver professional services to its customers. In turn, consultants from partners also needed to be trained on FDS products and services.

According to the then Training Manager, there was only one supplier that responded to the RFP. He found it to be a good LMS system which matched most of the requirements. On the other hand, he found it to be too expensive and therefore was not pursued (Table 8-2: EARLY_BP.2). In turn, the IT consulting company came up with Oracle iLearning as an alternative (Table 8-2: EARLY_SP.2). FDS Academy was given a trial account to try out Oracle iLearning (Table 8-2: EARLY_SS_F.1). The FDS Academy took this opportunity to create an online demo training environment adapted to its needs, i.e. using existing training materials and with the intention of further developing the demo into a real environment (Table 8-2: EARLY_BS.2). “After that, the rest is history”, said the manager. “We signed the hosting
contract with Oracle iLearning and implemented it.” (Table 8-2: EARLY_SP_3).

The next steps that were sent into motion to support the introduction of Oracle iLearning in FDS include staff training (Table 8-2: EARLY_SS_Q_1), and customizing the learning environment according to FDS look and feel (Table 8-2: EARLY_SS_Q_2).

The FDS Academy team began working on the development of e-learning modules that will be published in Oracle iLearning. As a start, the training modules consisted of learning units pertaining to the company procedures and policies such as leave forms, expenses declaration, resource reservations, and the like. It also expanded the scope of new employee training to include technical training about FDS products (Table 8-2: EARLY_BS_4). Basic training modules were developed also using the material from the existing class-room based training (See Section 7.2). These materials were intended for e-learning delivery through Oracle iLearning. For new employee training, the FDS Academy developed a one-day new employee training session consisting of a half-day introduction talks by the different departments and the rest of the day is spent on e-learning. Oracle iLearning was official introduced in FDS for this purpose (Table 8-2: EARLY_SP_3).

The use of Oracle iLearning quickly spread into the domain of classroom-based training, which was still being delivered purely as instructor-led instruction (Table 8-2: EARLY_BS_5). The FDS Academy came to the idea of delivering software product training in blended learning format for which Oracle iLearning is designed (Table 8-2: EARLY_SP_4). To realize this, the Oracle iLearning site must be revised to incorporate the product training curriculum. This also meant that the Powerpoint slides and other hand-outs used in the training should be published in Oracle iLearning (Table 8-2: EARLY_SS_F_3).

When FDS Academy signed up for Oracle iLearning, the hosting agreement is on a subscription basis. The FDS Academy pays for Oracle iLearning according to the number of users it enrolls in the software. This has to be known in advance before the contract is signed. Initially, FDS signed up for 100 users. It was also entitled to 5 MB of space per user. For the FDS Academy’s understanding, this means not enough space. It is a constraint (Table 8-2: EARLY_SS_F_4). Quite tacitly, employee accounts that were created did not have an expiration date. Only when the employee leaves the company, the Oracle iLearning account gets deleted (Table 8-2: EARLY_SS_F_7).

Working around the limitation in the host server space, the FDS Academy created its own in-house training server that Oracle iLearning can access (Table 8-2: EARLY_BS_6). In this training server, the training materials for classroom based training were structurally stored per course – the Powerpoint slides, theory and reading materials in pdf format, exercises in Excel, etc. The connection and access to this material was configured in Oracle iLearning in which links were made to the training hand-outs per course automatically (Table 8-2: EARLY_SS_F_6). With this set-up, revision and update of existing training materials did not involve working with Oracle iLearning. The work took place in the local training server (Table 8-2: EARLY_BS_7).

The local training server was hosted at the FDS’s training center. It was only accessible for in-house training sessions. This means that for onsite training delivery, the Oracle iLearning is not linked to the locally hosted materials. Onsite trainees were not able to view the hand-outs via Oracle iLearning (Table 8-2: EARLY_SS_F_8). This was one of the downside of the solution thought of in item (EARLY_BS_6). To mitigate this issue, onsite training participants were given USB sticks containing the training hand-outs. In this manner, they could access and view, for example, the Powerpoint slides during lecture (Table 8-2: EARLY_BS_8).

One specific requirement pertaining to training hand-outs in digital format is that they should be accessible to trainees only for the duration of the training session. In other words, the hand-outs for a course should be accessible online only while the course is on-going. The online accessibility of the pertinent training hand-outs should have a begin date and an expiry date
(Table 8-2: EARLY_SS_F_5).

Feedback though course evaluation forms an integral part of any training session. For training supported by a learning management system, it was expected Oracle iLearning should support filling in of online course evaluation forms (Table 8-2: EARLY_SS_F_9). On the other hand, Oracle iLearning did not have an explicit functionality called ‘Course evaluation’. Instead, it has a built-in ‘course assessment functionality’, which is intended for creating tests and quizzes. Having similar interaction processes, i.e. multiple choice exam vs. Likert scale rating options, the course assessment functionality could be used to create course evaluation forms (Table 8-2: EARLY_BS_9).

Another less desirable feature of Oracle iLearning was the output format of its reports. It was in a format that cannot be easily copied, pasted and distributed. Course evaluation results for example were displayed onscreen, i.e. web-based format and could be exported to other formats such as MS Excel without having knowledge of SQL (Table 8-2: EARLY_SS_F_10). To deal with this limitation, an alternative way of working was devised. The raw data onscreen were copied and pasted to MS Excel. Through Excel, the course evaluation results could be manipulated, i.e. sorted, categorized, visualized, etc., stored in portable format that could be attached to an e-mail and distributed to relevant stakeholders (Table 8-2: EARLY_BS_10).

As the FDS Academy sought to improve the quality of its training materials and to define what they wanted to achieve with training, the idea of certification and assessments came to mind. While the idea of certification was not realized, assessments were pushed through. The training development team would like to incorporate assessments in the blended-learning program, with each modules concluding with an assessment of test (Table 8-2: EARLY_BS_11).

With most of its learning materials online, the FDS Academy manager actively promoted the use of Oracle iLearning both for in-house and onsite training deliveries (Table 8-2: EARLY_SP_4). Whenever preparations were underway for an upcoming training, he reminded the training developer and coordinator to make sure that the trainer promotes the use of Oracle iLearning in the sessions (Table 8-2: EARLY_BS_14). Supporting instructional materials on how to use Oracle iLearning were added to the training hand-outs.

In the course of time, the FDS Academy had come up with certain policies, although not written, that defined each team member’s access rights to the Oracle iLearning. Being the most advanced user of Oracle iLearning, the FDS Academy manager became the site administrator with most rights. Eventually, this role was passed to the new training coordinator (discussed in Post Deployment). As training developers work mostly with content, they were given content management rights (Table 8-2: EARLY_BS_12). On the other hand, trainers were not given administrative right but were instead assigned as instructors (Table 8-2: EARLY_BS_13). This means that they cannot change content or enrollment but have special viewing privileges. This was to prevent immediate correction, update and publication of training materials by the trainer while the training is still on-going. Otherwise, versioning and document consistency will be an issue.

Finally, the FDS Academy added two new courses in the blended-learning curriculum: System Administration and Data Management Plus. It is therefore a requirement that these should be available in Oracle iLearning (Table 8-2: EARLY_SS_F_11).
### Table 8-2. Case D: Requirements at Early Implementation

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_SP_1</td>
<td>FDS Academy, tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_BP_1</td>
<td>FDS partners should also get training on FDS data solutions.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_BS_1</td>
<td>FDS Academy would like to engage the services of a consulting company for the LMS selection process. An RFP was sent out with a list of functional requirements. See Table 8-3. Requirements derived from RFP.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_BP_2</td>
<td>LMS Supplier who responded to the RFP was too expensive.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SP_2</td>
<td>Oracle iLearning proposed as affordable and suitable LMS alternative; FDS Academy should try out this product</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_F_1</td>
<td>FDS should have a trial account to try out Oracle iLearning.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_BS_2</td>
<td>FDS Academy should make use of trial account to create a demo environment that builds on existing training curriculum; this includes filling in template for site strategy, content, etc.</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_BS_3</td>
<td>FDS Academy wants to sign Oracle iLearning hosting contract.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_SS_Q_1</td>
<td>Staff should receive training in Oracle iLearning.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_SS_Q_2</td>
<td>Oracle iLearning LMS site must conform to FDS look and feel.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_BS_4</td>
<td>New employee training should also include a technical introduction to FDS products aside from company orientation. The technical training can be computer-based and conducted stand alone.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_BS_3</td>
<td>Oracle iLearning will be the default environment for delivering training in e-learning format; this is applicable especially to new employee training.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_BS_5</td>
<td>Classroom-based product training should also upgrade and innovate into blended learning</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning</td>
</tr>
<tr>
<td>15</td>
<td>EARLY_SS_F_3</td>
<td>Oracle iLearning should support multiple curricula, i.e. it should be possible to incorporate Powerpoint slides and exercises in pdf format used in classroom-based training.</td>
</tr>
<tr>
<td>16</td>
<td>EARLY_SS_F_4</td>
<td>Oracle iLearning subscription is limited to 100 users. For each user, FDS is entitled to 5 MB of space.</td>
</tr>
<tr>
<td>17</td>
<td>EARLY_SS_F_5</td>
<td>Hand-outs and lecture slides should be available in Oracle iLearning only for the duration of the training session. Trainee accounts should expire at the end of the training session.</td>
</tr>
<tr>
<td>18</td>
<td>EARLY_BS_6</td>
<td>Handouts and lecture slides should not be stored in the Oracle iLearning online server. They should be stored on a separate local server which interfaces with Oracle iLearning.</td>
</tr>
<tr>
<td>19</td>
<td>EARLY_SS_F_6</td>
<td>It should be possible within Oracle iLearning to link to materials in external servers and display them correctly.</td>
</tr>
<tr>
<td>20</td>
<td>EARLY_BS_7</td>
<td>Updating existing slides and materials should be done via the local training server.</td>
</tr>
<tr>
<td>21</td>
<td>EARLY_SS_F_7</td>
<td>Employee accounts do not expire.</td>
</tr>
<tr>
<td>22</td>
<td>EARLY_SS_F_8</td>
<td>The Oracle iLearning site for trainees at customer site will not contain powerpoint slides; however, they can view the training content outline.</td>
</tr>
<tr>
<td>23</td>
<td>EARLY_BS_8</td>
<td>For onsite training, trainees will have access to the training slides through USB sticks that will be provided for them.</td>
</tr>
<tr>
<td>24</td>
<td>EARLY_SS_F_9</td>
<td>Course evaluation should be done via Oracle iLearning. Trainees should be able to fill-in an online course evaluation form.</td>
</tr>
<tr>
<td>25</td>
<td>EARLY_BS_9</td>
<td>Assessment functionality in Oracle iLearning should be used to create course evaluation form.</td>
</tr>
<tr>
<td>26</td>
<td>EARLY_SS_F_10</td>
<td>Reporting in Oracle iLearning requires knowledge of databases. Results of course evaluation are in raw format requiring transformation into other formats distributable per email.</td>
</tr>
<tr>
<td>27</td>
<td>EARLY_BS_10</td>
<td>Use MS Excel in processing raw course evaluation results, for distribution and archiving.</td>
</tr>
<tr>
<td>28</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
</tr>
<tr>
<td>29</td>
<td>EARLY_BS_12</td>
<td>The FDS Academy should be the system and site administrator of the FDS Oracle iLearning site. The training developer has content management rights.</td>
</tr>
<tr>
<td>30</td>
<td>EARLY_BS_13</td>
<td>The trainers do not have administrative rights, i.e. enrolment rights and content management rights. Trainer will have instructor rights (which mean no administrative rights, i.e. enrolment and content management).</td>
</tr>
<tr>
<td>31</td>
<td>EARLY_BS_14</td>
<td>The trainer should promote the use of Oracle iLearning in training sessions.</td>
</tr>
<tr>
<td>32</td>
<td>EARLY_SS_F_11</td>
<td>Two new courses are added in the FDS curriculum. This should also be incorporated in the Oracle FDS iLearning site.</td>
</tr>
</tbody>
</table>
Table 8.3. Case D: Requirements at Early Implementation – RFP written requirements

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_SS_Q_3</td>
<td>[RFP1] A hosted-solution, so that the company is not required to tax internal IT infrastructures to support the roll-out of the FDS Academy.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SS_Q_4</td>
<td>[RFP2] Option of potentially relocating the solution to an internal web server in the future as demand increases.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SS_F_12</td>
<td>[RFP3] Prefer an SQL database format (to be compliant with internal systems).</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SS_F_13</td>
<td>[RFP4] The company will give strong consideration to those vendors that have built-in authoring capability or can recommend a seamless content authoring approach. Asset Control is additionally considering the use of Trivantis, Lectora Publisher for content authoring and is still seeking a solution for rapid creation of software application simulations (i.e. RapidBuilder from XStream Software or OnDemand from Global Knowledge).</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SS_F_14</td>
<td>[RFP5] Performance tracking for asynchronous course material (using industry standards).</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_Q_5</td>
<td>[RFP6] A strong security module, allowing for appropriate access to course content and administrative data.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_SS_F_15</td>
<td>[RFP7] Threaded Discussion Groups (preferably linked to specific courses).</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_SS_F_16</td>
<td>[RFP8] Ability to attach PowerPoint and Word Documents as reference material.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_SS_F_17</td>
<td>[RFP9] Integrated email collaboration (learner to instructor, learner-to-learner, etc.).</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_Q_6</td>
<td>[RFP11] 24X7 support.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_SS_F_20</td>
<td>[RFP13] Level-1 Evaluation Utility (creating “happy sheets”).</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_SS_Q_7</td>
<td>[RFP14] Regulatory Compliance.</td>
</tr>
<tr>
<td>16</td>
<td>EARLY_SS_F_22</td>
<td>[RFP16] Doesn’t require a plug-in.</td>
</tr>
<tr>
<td>17</td>
<td>EARLY_SS_F_23</td>
<td>[RFP17] Classroom management (may use in the future, but not part of initial implementation)</td>
</tr>
<tr>
<td>19</td>
<td>EARLY_SS_F_25</td>
<td>[RFP19] Competency Management (future).</td>
</tr>
<tr>
<td>20</td>
<td>EARLY_SS_F_26</td>
<td>[RFP20] 360-degree evaluation (preferred for future).</td>
</tr>
<tr>
<td>22</td>
<td>EARLY_SS_Q_8</td>
<td>[RFP22] Even though FDS is located in the Netherlands, all training sessions will be conducted in English. There is no need for multi-lingual support or localization.</td>
</tr>
<tr>
<td>23</td>
<td>EARLY_SS_Q_9</td>
<td>[RFP23] The company has no immediate plans for providing e-commerce support (credit card transactions) in the initial implementation.</td>
</tr>
<tr>
<td>24</td>
<td>EARLY_SS_Q_10</td>
<td>[RFP24] There is no requirement at this time to integrate performance data with an external ERP or CRM system. This should be a fairly straightforward implementation of your hosted-LMS offering.</td>
</tr>
</tbody>
</table>

8.4 Requirements at post-deployment: buy out and the financial crisis

In 2006, FDS Academy expanded its team as the demand for onsite training continues to grow. For that year, it delivered 29 training sessions, 60% more than the previous year. Each training session lasted for a minimum of 3 days to a maximum of 9 consecutive working days. New employee training also became more frequent. In the year that followed, onsite training deliveries doubled and the US became a popular training site for US-based clients. In the same year (2007),
FDS was bought by its biggest client. This buy-out resulted in major organizational restructuring that affected almost departments within FDS, including FDS Academy. Finally in 2008 when the global financial crisis struck, FDS was badly affected. This led to drastic decisions that resulted in the dissolution of the training department.

In this phase, we touch on these developments and try to provide account on how these influence the requirements for Oracle iLearning. The requirements are shown in Table 8-4.

Table 8-4. Case D: Requirements at Post Deployment

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POST_SS_F_1</td>
<td>User subscription should be increased to 135 users.</td>
</tr>
<tr>
<td>2</td>
<td>POST_SS_Q_1</td>
<td>Training materials, including the FDS Oracle iLearning site should be consistent with the new housestyle.</td>
</tr>
<tr>
<td>3</td>
<td>POST_BP_1</td>
<td>Increasingly, Sales and US-based account managers continue to request for onsite training to customers who didn’t sign a software license agreement yet. Expressed knowledge gap by customers on FDS software is seen as a training need.</td>
</tr>
<tr>
<td>4</td>
<td>POST_BS_1</td>
<td>The new training coordinator should have administrative rights. Each training session should be set-up by the training coordinator in Oracle iLearning.</td>
</tr>
<tr>
<td>5</td>
<td>POST_BS_2</td>
<td>Customers and partners can make use of the e-learning modules in FDS as an interim solution for a training session that will come later.</td>
</tr>
<tr>
<td>6</td>
<td>POST_SP_1</td>
<td>Oracle iLearning should be seen as a sales and account management support tool.</td>
</tr>
<tr>
<td>7</td>
<td>POST_BP_2</td>
<td>Training session set-up is increasingly becoming tiresome. Oracle iLearning performance issues are becoming more prevalent. Downtimes are getting frequent; there is almost no training session setup with no down time or error messages received while uploading new materials.</td>
</tr>
<tr>
<td>8</td>
<td>POST_BP_3</td>
<td>Negative course feedback received as trainee access to learning content, i.e. a pdf file or slide is getting slower.</td>
</tr>
<tr>
<td>9</td>
<td>POST_BP_4</td>
<td>Quality of training deteriorates due time outs during onsite training. Trainer reports that the course evaluation takes too long to load and often leads to time outs.</td>
</tr>
<tr>
<td>10</td>
<td>POST_BS_3</td>
<td>Training coordinator would like to migrate course evaluation form after seeing the Marketing Department’s online survey environment.</td>
</tr>
<tr>
<td>11</td>
<td>POST_SP_2</td>
<td>Course evaluation doesn’t have to take place in Oracle iLearning anymore.</td>
</tr>
<tr>
<td>12</td>
<td>POST_BS_4</td>
<td>Need for office space in home office prompts training center relocation to Amsterdam office.</td>
</tr>
<tr>
<td>13</td>
<td>POST_BS_5</td>
<td>FDS Academy will become FDS Training. FDS Training becomes part of beta-testing department, Solution Center. FDS Training is split into two locations: 2 team members remain in home office and 2 team members plus the manager work at the Amsterdam office.</td>
</tr>
<tr>
<td>14</td>
<td>POST_BP_5</td>
<td>New FDS training manager wants to cut down costs.</td>
</tr>
<tr>
<td>15</td>
<td>POST_SS_F_2</td>
<td>Oracle iLearning subscription should be reduced to 100 users.</td>
</tr>
<tr>
<td>16</td>
<td>POST_SS_F_3</td>
<td>Employee accounts should expire. Accounts of those who have already undergone new employee training should be deleted.</td>
</tr>
<tr>
<td>17</td>
<td>POST_BP_6</td>
<td>New members of the team, i.e. the new training developer finds Oracle iLearning a difficult system to use. Even after having an intensive orientation, she finds training setup tasks in Oracle to be too complicated.</td>
</tr>
<tr>
<td>18</td>
<td>POST_BP_7</td>
<td>After so many tries, the new training developer cannot get an assessment up and running in Oracle iLearning.</td>
</tr>
<tr>
<td>19</td>
<td>POST_BP_8</td>
<td>User help and how-to tutorials, especially in setting up assessments, seem to be unavailable.</td>
</tr>
<tr>
<td>20</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>21</td>
<td>POST_BS_6</td>
<td>The search for an open source LMS should be assigned to an intern. The intern can also make an inventory of LMS requirements for FDS Training. See Table 8-5 for the open LMS requirements inventory.</td>
</tr>
<tr>
<td>22</td>
<td>POST_BP_9</td>
<td>Due to a lack of clear direction and manager commitment, the intern left the open source LMS project. Oracle iLearning will not be replaced in the meantime.</td>
</tr>
<tr>
<td>23</td>
<td>POST_BP_10</td>
<td>FDS is not able to sign any license agreement in the last 6 months of 2008. The prognosis for next year is zero sales. Customers are canceling maintenance and support agreements.</td>
</tr>
<tr>
<td>24</td>
<td>POST_BS_7</td>
<td>To cut down costs and remain viable. FDS has to reduce its staff. The training department has to be dissolved.</td>
</tr>
<tr>
<td>25</td>
<td>POST_SP_4</td>
<td>With the dissolution of the training department, Oracle iLearning is no longer needed. Subscription has to be at minimum to meet contractual obligation with customers.</td>
</tr>
<tr>
<td>26</td>
<td>POST_SP_5</td>
<td>Other e-learning authoring tool, i.e. Adobe Captivate should be used for creating assessments instead of Oracle iLearning.</td>
</tr>
</tbody>
</table>

POST_SS_F_1. “User subscription should be increased to 135 users.” One of the very first things that needed change with respect to Oracle iLearning was the number of subscribed users.
Specifically, the manager wanted to increase the number of subscribed users to 135, based on conservative estimates of future growth. The fixed user base in the LMS was growing not only because of increasing personnel, but also because the LMS has found another use. Oracle iLearning was not only being used to support blended learning and new employee training, but it had also become a sales and account management support tool (Table 8-4: POST_SP_1). We will explain this in the following accounts.

Going back to pre-implementation, training was established because customers who bought the software license badly felt the need for know-how in implementing and using FDS data management tools. Training, therefore, was intended for helping those customers who already have FDS installed. Over the years, training was viewed from the point of view of sales and potential customers as a post-installation knowledge transfer session to a pre-implementation workshop. The FDS Academy had observed that US-based sales personnel and account managers often pushed for onsite training for their potential customers (Table 8-4: POST_BP_1). These colleagues inquired and requested training from the FDS Academy for delivery to third-parties who were not customers yet. This was a big issue for FDS Academy not only because of its limited capacity but also because it was not convinced that training is the solution for the potential customer’s articulation of its knowledge gap on FDS products. In most instances, the potential customer doesn’t know yet what it wants and had a vague idea of what its knowledge gaps are. For example, a sales specialist contacted the training coordinator for training onsite and when asked for more details like in which areas of FDS data management solution does the client need training, the response was “they are basically asking us for guidance on what to do”. In situations like this, the FDS Academy felt that Sales was not doing its job of properly screening customer needs and was passing on the task. Although it was not always Sales who misinterpret the customers’ knowledge gaps, sometimes the misinterpretation came from the customers themselves. Any knowledge gap about the product was articulated as a training need.

The issue generated a lot of discussion within the team and it escalated to a level higher-up in the hierarchy. One of the conclusions and suggestion put forward to address the issue was to enable potential customers and customers who cannot wait for training to log-in to the eLearning in Oracle iLearning (Table 8-4: POST_SP_1). The eLearning site hosted basic FDS modules and demos that could serve as training preview for a face-to-face training that will come later. It could also provide an overview and a mental model of what FDS is, what its interface looks like, what artifacts in the system can the user recognize in relation to his or her work, etc.

POST_SS_Q_1. “Training materials, including the FDS Oracle iLearning site should be consistent with the new housestyle.” FDS paid a lot of attention to its marketing representation. When a new house style was introduced for documents, logos and marketing materials including its website was released, the Oracle iLearning also needed to follow suit. As slightly discussed in early implementation, user roles also changed in Oracle iLearning. The new training coordinator became FDS Oracle iLearning site administrator. Some of the tasks performed by the FDS Academy manager relating to training were passed on to the coordinator so that the former can focus on documentation tasks (Table 8-4: POST_BS_1).

In this later period, five years since Oracle iLearning has been implemented, signs of decline began to show. Performance issues became more prevalent. Down times were getting frequent, especially during training set-up configuration. There was almost no training session setup with no down time or error messages received while configuring and compiling the modules for a session (Table 8-4: POST_BP_2). For the users’ part, access to learning content such as the slides or readings in pdf format was also getting slower. This led to negative course feedback from trainees (Table 8-4: POST_BP_3). On certain occasions it took minutes for a page to load. There were also instances, usually, after an onsite training delivery, that the course evaluation form was not filled-in. Sometimes this happened due to the trainer: he sometimes forgot to ask participants to fill-in the form. However, in most instances, the trainer reported that the course evaluation
form was not filled in because the form takes too long to load and often leads to time-outs (Table 8-4: POST_BP_4).

With this problem in mind, one time while the training coordinator visited a colleague for a chat in the marketing department, she saw the Marketing Programs Manager working with a survey form. She discovered that the Marketing Department regularly makes use of surveys for customers and especially during the annual End-user Conference. The surveys were done through a specialized online survey package called Zapsurvey. The marketing manager said that the survey was not optimally being used, i.e. used only once to twice per year, and it can contain multiple surveys. He recommended using it. The training coordinator tested the online web-based survey and from there she migrated the course evaluation form from Oracle iLearning to Zapsurvey (Table 8-4: POST_BS_3). From this onwards, course evaluation now took place in Zapsurvey. It didn’t take place in Oracle iLearning anymore (Table 8-4: POST_SP_2).

The continuous growth in staffing in the home office also meant that working space should be available to new employees. The need for office space and how to create it had been an on-going discussion within FDS. Several managers had identified the training room as an ideal office space that can house a small team. As a replacement, it was suggested that the training center should relocate to the Amsterdam office because more space was available there (Table 8-4: POST_BS_4).

In this phase, a major event took place: FDS was bought by its largest customer. This customer has the biggest software license agreement with FDS, and on its own is a large privately-held investment company based in the US. This event triggered a series of organizational changes such as a new management board, a new management layer – VPs (vice-presidents), a shift in home office location with New York as the management headquarters, and internal restructuring and alignments among different organizational units. The FDS Academy was appended to the beta-testing department called Solution Center, which is based in the Amsterdam office. The original FDS Academy minus its original manager became FDS Training and the Solution Center manager became its new head. This change in the training department was triggered by the VP of Services who believed that the training department can benefit from the expertise of Solution Center as a beta-testing group. As a beta-testing group, Solution Center is structured like a surrogate FDS customer that simulates the functions and processes of a financial institution. It has its own set of business analysts, IT professionals and access to data resources such as vendor feed data. The group also serves an internship ground for new professional services consultants. Sometimes, the IT professionals in the group carry tasks as professional services consultants. In this manner, the training department as part of Solution Center can have access to a pool of potential trainers. This change eventually facilitated the relocation of the training center and a few of its team members to Amsterdam. Two members of the team remained in the Friesland office and the two new members plus the new manager work in Amsterdam (Table 8-4: POST_BS_5).

One of the things that the new training manager would like to implement was to cut down costs (Table 8-4: POST_BP_5). Whether there was an official top management directive to cut down on costs was not clear. For the original FDS Training team members, especially the original ones, this was a not well-founded objective. As a service-producing department, the training team has been a viable department within FDS because it generates income through the delivery of training services. On the other hand, the Solution Center on its own does not generate income.

Several cost areas were identified for economizing: travel and accommodation including Oracle iLearning. To begin with, the new manager was very interested in migrating to an open source LMS system (Table 8-4: POST_SP_3). He believed that such free systems could reduce annual costs of LMS subscription. Since migration would not happen immediately, small steps to reduce costs were being pushed forward. For example, the new manager would have liked to reduce the number of subscribed users from 135 to 100 (Table 8-4: POST_SS_F_2). To free up
some accounts, it was also suggested that employee accounts should expire, and those accounts that were inactive should be deleted (Table 8-4: POST_SS_F_3). However, there were accounts that FDS Training could not delete. FDS was bound by contractual obligations to some of its customers, and more importantly to the mother company that bought it, which had rights to unlimited access to the eLearning modules.

In the mean time, the newer members of the team were becoming more articulate about the difficulties they encounter when working with Oracle iLearning. For example, the new training developer complained about not being able to understand the way of working with the system. Even after having an intensive orientation with the senior training developer, she was not able to accomplish an end-to-end task, say modifying the curriculum in Oracle iLearning to incorporate new modules, on her own (Table 8-4: POST_BP_6). After so many tries, she was not able to get an assessment up and running in Oracle iLearning (Table 8-4: POST_BP_7). She also pointed out that user help and how-to tutorials, especially in setting up assessments, were not available (Table 8-4: POST_BP_8). Due to issues in creating assessments in Oracle iLearning, it was further suggested that assessments, exams and quizzes should instead be developed separately in other eLearning authoring tools such as Adobe Captivate (Table 8-4: POST_SP_5).

These known issues together with the desire to cut down on costs were used to justify the move to a new open source LMS. For this effort, FDS training utilized a student intern who was assigned to gather requirements from the team and to research on available open source LMS that can meet its needs (Table 8-4: POST_BS_6). The student intern drew up an inventory of requirements based mostly on discussions with the manager and the rest of the team working in Amsterdam. A verbatim representation of these requirements is given in Table 8-5. These requirements appeared not to have the support of the team working in the Friesland office. This was reflected in how things went during virtual team meetings. Some members were very critical about the requirements, stating that these were not unique requirements and have rhetorically argued that Oracle iLearning sufficiently meets the requirements, therefore there was no need for a new one.

It was obvious within the training department that the relationship with the manager and the training team was constrained. This was one of the side effects of the organizational change. Some of them were very articulate about their dislike of the manager. With unclear project objectives, a singular vision, lack of supervision and committed support from the manager, the intern left the project (Table 8-4: POST_BP_9). This meant that Oracle iLearning will not be replaced in the meantime.

In the beginning of the second half of 2008, a global financial meltdown was on its way to herald a crisis. FDS was severely affected. It was not able to generate sales in a span of six months. The prognosis was zero sales for the following year. Customers began calling to cancel maintenance and support agreements (Table 8-4: POST_BP_10). It was panic time.

As a natural reaction, in order to remain viable, FDS had already embarked a massive and abrupt personnel reduction measures. With a projection of zero software license sales for the following year, training services can then be expected to be null. With this pessimistic view of the world, the training department within FDS was dissolved, its team members, except the manager, were let go (Table 8-4: POST_BS_7).

With the dissolution of the training department, Oracle iLearning was no longer felt needed (Table 8-4: POST_SP_4). Therefore, the subscription for next year needed to be reduced to the minimum in which FDS also meets its contractual obligations with its customers. Such action is tantamount to abandonment.
Table 8-5. Case D: Requirements at Post Deployment – New Requirements for Open Source LMS

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>POST_SS_F.5</td>
<td>[OpenSrcLMS] Course delivery: PDF’s usable (nice to have).</td>
</tr>
<tr>
<td>3</td>
<td>POST_SS_F.6</td>
<td>[OpenSrcLMS] Course delivery: Place to store assignments and reading material per lesson/module.</td>
</tr>
<tr>
<td>4</td>
<td>POST_SS_F.7</td>
<td>[OpenSrcLMS] Course delivery: Course calendar, preferably smoothly looking.</td>
</tr>
<tr>
<td>11</td>
<td>POST_SS_F.14</td>
<td>[OpenSrcLMS] Course delivery ➤ Assessments: Grading of coursework and roster processing, including waitlisting.</td>
</tr>
<tr>
<td>12</td>
<td>POST_SS_F.15</td>
<td>[OpenSrcLMS] Course delivery ➤ Assessments: Assessments usable, preferably to import via SCORM /QTI.</td>
</tr>
<tr>
<td>13</td>
<td>POST_SS_F.16</td>
<td>[OpenSrcLMS] Course delivery ➤ Assessments: Give feedback on tests.</td>
</tr>
<tr>
<td>14</td>
<td>POST_SS_F.17</td>
<td>[OpenSrcLMS] Course delivery ➤ Assessments: Create reports on achievements of students on tests.</td>
</tr>
<tr>
<td>15</td>
<td>POST_SS_F.18</td>
<td>[OpenSrcLMS] Course delivery ➤ Create curriculum from separate modules/courses.</td>
</tr>
<tr>
<td>16</td>
<td>POST_SS_F.19</td>
<td>[OpenSrcLMS] Course delivery: Create curriculum from separate modules/courses.</td>
</tr>
<tr>
<td>18</td>
<td>POST_SS_F.21</td>
<td>[OpenSrcLMS] Course delivery: Ability to create an index so that people can find a particular topic.</td>
</tr>
<tr>
<td>20</td>
<td>POST_SS_F.23</td>
<td>[OpenSrcLMS] Course delivery: Use of recorded video and audio?</td>
</tr>
<tr>
<td>21</td>
<td>POST_SP_6</td>
<td>[OpenSrcLMS] Course delivery: Training portal idea?</td>
</tr>
<tr>
<td>22</td>
<td>POST_SS_F.24</td>
<td>[OpenSrcLMS] Course delivery: Search functionality for a particular LU</td>
</tr>
<tr>
<td>23</td>
<td>POST_SS_F.25</td>
<td>[OpenSrcLMS] Management: Manage users and groups</td>
</tr>
<tr>
<td>24</td>
<td>POST_SS_F.26</td>
<td>[OpenSrcLMS] Management: Minimal 200 user accounts needed (over 130 in use now)</td>
</tr>
<tr>
<td>26</td>
<td>POST_SS_F.28</td>
<td>[OpenSrcLMS] Management: Provision for external parties to book training (nice to have)</td>
</tr>
<tr>
<td>27</td>
<td>POST_SS_F.29</td>
<td>[OpenSrcLMS] Management: Use of grouping to allow students access to specific modules or curricula</td>
</tr>
<tr>
<td>28</td>
<td>POST_SS_F.30</td>
<td>[OpenSrcLMS] Management: Be able to add modules/lessons to curricula</td>
</tr>
<tr>
<td>29</td>
<td>POST_SS_F.31</td>
<td>[OpenSrcLMS] Management: Generate reports (on users (amount of logins per month) user’s logon time, etc)</td>
</tr>
<tr>
<td>30</td>
<td>POST_SS_F.32</td>
<td>[OpenSrcLMS] Management: Multiple teachers per course – one single teachers account?</td>
</tr>
<tr>
<td>31</td>
<td>POST_SS_F.33</td>
<td>[OpenSrcLMS] Management: Auto enrollment, with coordinators confirmation (enrolling learners in courses when required according to predefined criteria, such as job title or work location)</td>
</tr>
<tr>
<td>33</td>
<td>POST_SS_F.35</td>
<td>[OpenSrcLMS] Communication / Interactivity: Discussion forums?</td>
</tr>
<tr>
<td>36</td>
<td>POST_SP_7</td>
<td>[OpenSrcLMS] Technological: What to do with Dickens? meaning what is the implication to the existing training environment separate from Oracle Learning.</td>
</tr>
<tr>
<td>37</td>
<td>POST_SS_Q.3</td>
<td>[OpenSrcLMS] Technological: Adaptable interface to company image</td>
</tr>
<tr>
<td>38</td>
<td>POST_SS_Q.4</td>
<td>[OpenSrcLMS] Technological: No extra plugins needed, perhaps only flash or silverlight.</td>
</tr>
<tr>
<td>41</td>
<td>POST_SS_Q.7</td>
<td>[OpenSrcLMS] Other: Solid structure of pages in the system.</td>
</tr>
<tr>
<td>42</td>
<td>POST_SS_C.1</td>
<td>[OpenSrcLMS] Other: Not more expensive than Oracle.</td>
</tr>
<tr>
<td>43</td>
<td>POST_SS_Q.8</td>
<td>[OpenSrcLMS] Other: Fast interface (not as slow as Oracle) i.e. it should take less than n … seconds to load a new slide / … to load a demo.</td>
</tr>
</tbody>
</table>

8.5 Within-domain evolution

The dynamics of how each requirements domain evolves over time based on this case are
discussed in the following sections. Specifically, attention is given to requirements instances in each domain that influence other requirements in the same domain.

8.5.1 Business problem domain evolution

The business problem domain in this case is a closely constrained system of problem statements that revolve around a singular, well-articulated business need, namely the need for training. Beginning with the identification of customer needs in the pre-implementation phase, the business problem domain has evolved from an external driven push towards internal capacity building, i.e., organizational structuring such as creating a training department, to tooling in the form of a learning management system. Ultimately, the business problems revert to external focus with the onslaught of the global financial crisis. Table 8-6 is a compilation of the business problem requirements gathered in this case.

Table 8-6: Business problem statements compilation: all phases

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BP_1</td>
<td>Aside from consulting services, customers demand for more structured and formal knowledge transfer efforts such as training and documentation regarding the use and implementation of FDS software.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BP_2</td>
<td>For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BP_3</td>
<td>The new training department is not experienced with the provision of training services and development of training materials.</td>
</tr>
<tr>
<td>4</td>
<td>PRE_BP_4</td>
<td>Training handouts should be available and accessible to trainees during and after the training.</td>
</tr>
<tr>
<td>5</td>
<td>PRE_BP_5</td>
<td>Customers would like to have training conducted in their own offices.</td>
</tr>
<tr>
<td>6</td>
<td>PRE_BP_6</td>
<td>FDS feels the need to bring (new) employees up to speed in term of knowledge and competency.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_BP_1</td>
<td>FDS partners should also get training on FDS data solutions.</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_BP_2</td>
<td>LMS Supplier who responded to the RFP was too expensive.</td>
</tr>
<tr>
<td>9</td>
<td>POST_BP_1</td>
<td>Increasingly, Sales and US-based account managers continue to request for onsite training to customers who didn’t sign a software license agreement yet. Expressed knowledge gap by customers on FDS software is seen as a training need.</td>
</tr>
<tr>
<td>10</td>
<td>POST_BP_2</td>
<td>Training session set-up is increasingly becoming toilsome. Oracle iLearning performance issues are becoming more prevalent. Downtimes are getting frequent; there is almost no training session setup with no down time or error messages received while uploading new materials.</td>
</tr>
<tr>
<td>11</td>
<td>POST_BP_3</td>
<td>Negative course feedback received as trainee access to learning content, i.e. a pdf file or slide is getting slower.</td>
</tr>
<tr>
<td>12</td>
<td>POST_BP_4</td>
<td>Quality of training deteriorates due time outs during onsite training. Trainer reports that the course evaluation takes too long to load and often leads to time outs.</td>
</tr>
<tr>
<td>13</td>
<td>POST_BP_5</td>
<td>New FDS training manager wants to cut down costs.</td>
</tr>
<tr>
<td>14</td>
<td>POST_BP_6</td>
<td>Newer members of the team, i.e. the new training developer finds Oracle iLearning a difficult system to use. Even after having an intensive orientation, she finds training setup tasks in Oracle to be too complicated.</td>
</tr>
<tr>
<td>15</td>
<td>POST_BP_7</td>
<td>After so many tries, the new training developer cannot get an assessment up and running in Oracle iLearning.</td>
</tr>
<tr>
<td>16</td>
<td>POST_BP_8</td>
<td>User help and how-to tutorials, especially in setting up assessments, seem to be unavailable.</td>
</tr>
<tr>
<td>17</td>
<td>POST_BP_9</td>
<td>Due to a lack of clear direction and manager commitment, the intern left the open source LMS project. Oracle iLearning will not be replaced in the meantime.</td>
</tr>
<tr>
<td>18</td>
<td>POST_BP_10</td>
<td>FDS is not able to sign any license agreement in the last 6 months of 2008. The prognosis for next year is zero sales. Customers are canceling maintenance and support agreements.</td>
</tr>
</tbody>
</table>

The business problem domain together with the software product concept domain is the least represented requirements domain in this case. While there are no interdependencies between the business problem statements in the form of an impact relation, there are, however, associations
among problem statements. We can see this in items 2 & 3 in Table 8-6. In this example, the statements PRE_BP_2. “For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.” and PRE_BP_3. “The new training department is not experienced with the provision of training services and development of training materials.” are problem statements of the same mold. They are problem articulations of a shared larger problem. In other words, they share the same problem class, which is in this case is the limited capacity of the new training department in terms of know-how and resources to carry out its needed role in the organization.

8.5.2 Business solution domain evolution

Requirements in the form of business solution specification are very common in this case (Table 8-7). They are most evident in the early implementation phase. This is when the LMS project is begun. It culminates in the choice for Oracle iLearning as the LMS tool that will support the team in carrying out its planned training activities and strengthen its training role. The statements include policies on how to use the system such as the delineating roles in the system and who in the team will have what role in Oracle iLearning. They are also rife with identifications of opportunity areas where Oracle iLearning can be used. For example, the statements represented by EARLY_BS_4 and EARLY_BS_5 (Table 8-7: Items 12 and 13) identify new employee training and classroom-based product training as functional areas where Oracle iLearning can be used. They also provide specific ways of working around with the system such as overcoming issues with onsite training (see Table 8-7: Item 16) by using USB sticks to contain handouts.

During early implementation, the business solution domain evolved to have a software focus, identifying functional learning tasks that need to be carried out in the system. Meanwhile, in the phase before this, the business solution statements reflect the preparations being done to setup the training department and in getting it ready to deliver training services. Mostly, the statements are about capability building efforts advocating organizational and process change but without software. While software packages are alluded in the use of training PCs (Table 8-1: Item 9, PRE_BS_5), they are however givens and cannot be considered as software solutions. They can be referred as digital learning artifacts together with the USB sticks and training PCs. In addition, even though statements do touch upon functional aspects of learning and a few digital artifacts, they do not however reflect an awareness of an integrated software solution in the form of an LMS. The shift from non-software based solutions towards a software-focused solution is a feature the business solution domain evolution in this case. Another dominant theme reflected by the business solution domain is the fast-paced organizational expansion in terms of staffing and in the latter phases of implementation a sudden crimp. The changing dynamics of the organization, especially the takeover and its implications are also reflected in the post-deployment period business solution statements. The expansion and decline in staffing are captured in statements such as POST_BS_5 (Table 8-7: Item 27) and POST_BS_7 (Table 8-7: Item 29). They also foreshadow the increasing interest in other software solutions aside from Oracle iLearning. This is because Oracle iLearning was showing signs of failure in performance and usability (see Table 8-5, software solutions specification statements) and of equal importance, other managerial goals that were less interested in an LMS. Measures were taken to address some of the shortcomings of Oracle iLearning. For example the course evaluation form was moved from Oracle iLearning to another web-based service (Table 8-7: Item 25, POST_BS_3). Therefore, we can describe the business solution domain evolution in this case as shift from non-software-based measures towards LMS-focused efforts, specifically for Oracle iLearning and then a move away from Oracle iLearning towards other software solutions. The latter for example is captured in the statement POST_BS_6. “The search for an open source LMS should be assigned to an intern. The intern can also make an inventory of LMS requirements for FDS Training. See Table 8-5 for
the open LMS requirements inventory.” (Table 8-7: Item 28).

Table 8-7. Business solution statements compilation: all phases

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRE_BS_1</td>
<td>To address customer demands, FDS should set-up a training and documentation department: FDS Academy.</td>
</tr>
<tr>
<td>2</td>
<td>PRE_BS_2</td>
<td>Existing resources should be used i.e. in-house technical writer to lead the creation and set-up of the training department.</td>
</tr>
<tr>
<td>3</td>
<td>PRE_BS_3</td>
<td>Consultants with deep FDS system knowledge should become trainers and the vacant room in the building can become a training lab.</td>
</tr>
<tr>
<td>4</td>
<td>PRE_BS_4</td>
<td>Assistance of third party experts can be used to help develop the AC Academy training curriculum.</td>
</tr>
<tr>
<td>5</td>
<td>PRE_BS_5</td>
<td>Training will be class-room based, instructor-led training.</td>
</tr>
<tr>
<td>6</td>
<td>PRE_BS_6</td>
<td>For the training lab, there should be one PC per trainee.</td>
</tr>
<tr>
<td>7</td>
<td>PRE_BS_7</td>
<td>Trainees will receive the training materials in electronic format on USB stick but excluding the FDS software</td>
</tr>
<tr>
<td>8</td>
<td>PRE_BS_8</td>
<td>As business opportunities grow, FDS should create new functions and hire more staff.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_BS_1</td>
<td>FDS Academy would like to engage the services of a consulting company for the LMS selection process. A RFP was sent out with a list of functional requirements. See Table 8-3. Requirements derived from RFP.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_BS_2</td>
<td>FDS Academy should make use of trial account to create a demo environment that builds on existing training curriculum; This includes filling in template for site strategy, content, etc.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_BS_3</td>
<td>FDS Academy wants to sign Oracle iLearning hosting contract.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_BS_4</td>
<td>New employee training should also include a technical introduction to FDS products aside from company orientation. The technical training can be computer-based and conducted stand alone.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_BS_5</td>
<td>Classroom-based product training should also upgrade and innovate into blended learning.</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_BS_6</td>
<td>Handouts and lecture slides should not be stored in the Oracle iLearning online server. They should be stored on a separate local server which interfaces with Oracle iLearning.</td>
</tr>
<tr>
<td>15</td>
<td>EARLY_BS_7</td>
<td>Updating existing slides and materials should be done via the local training server.</td>
</tr>
<tr>
<td>16</td>
<td>EARLY_BS_8</td>
<td>For onsite training, trainees will have access to the training slides through USB sticks that will be provided for them.</td>
</tr>
<tr>
<td>17</td>
<td>EARLY_BS_9</td>
<td>Assessment functionality in Oracle iLearning should be used to create course evaluation form.</td>
</tr>
<tr>
<td>18</td>
<td>EARLY_BS_10</td>
<td>Use MS Excel in processing raw course evaluation results, for distribution and archiving.</td>
</tr>
<tr>
<td>19</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
</tr>
<tr>
<td>20</td>
<td>EARLY_BS_12</td>
<td>The FDS Academy should be the system and site administrator of the FDS Oracle iLearning site. The training developer has content management rights.</td>
</tr>
<tr>
<td>21</td>
<td>EARLY_BS_13</td>
<td>The trainers do not have administrative rights, i.e. enrolment rights and content management rights. Trainer will have instructor rights (which mean no administrative rights, i.e. enrolment and content management).</td>
</tr>
<tr>
<td>22</td>
<td>EARLY_BS_14</td>
<td>The trainer should promote the use of Oracle iLearning in training sessions.</td>
</tr>
<tr>
<td>23</td>
<td>POST_BS_1</td>
<td>The new training coordinator should have administrative rights. Each training session should be set-up by the training coordinator in Oracle iLearning.</td>
</tr>
<tr>
<td>24</td>
<td>POST_BS_2</td>
<td>Customers and partners can make use of the e-learning modules in FDS as an interim solution for a training session that will come later.</td>
</tr>
<tr>
<td>25</td>
<td>POST_BS_3</td>
<td>Training coordinator would like to migrate course evaluation form after seeing the Marketing Department’s online survey environment.</td>
</tr>
<tr>
<td>26</td>
<td>POST_BS_4</td>
<td>Need for office space in home office prompts training center relocation to Amsterdam office.</td>
</tr>
<tr>
<td>27</td>
<td>POST_BS_5</td>
<td>FDS Academy will become FDS Training. FDS Training becomes part of beta-testing department, Solution Center. FDS Training is split into two locations: 2 team members remain in home office and 2 team members plus the manager work at the Amsterdam office.</td>
</tr>
<tr>
<td>28</td>
<td>POST_BS_6</td>
<td>The search for an open source LMS should be assigned to an intern. The intern can also make an inventory of LMS requirements for FDS Training. See Table 7-5 for the open LMS requirements inventory.</td>
</tr>
<tr>
<td>29</td>
<td>POST_BS_7</td>
<td>To cut down costs and remain viable, FDS has to reduce its staff. The training department has to be dissolved.</td>
</tr>
</tbody>
</table>

As observed in the last three cases, there are also observable relationships between business
solutions in this case. Some business solution statements are related to other business solution statements because these reinforce or articulate them. The following instances illustrate this.

- **PRE_BS_2.** “Utilize existing resources, i.e. in-house technical writer to lead the creation and set-up of the training department.” → **PRE_BS_3:** “Consultants with deep FDS system knowledge should become trainers and the vacant room in the building can become a training lab.”

These two statements are related to each other in a way that the latter is a means of implementing the former. Utilizing consultants as trainers and making use of existing space in the office are all means of maximizing existing resources in order build up the capacity of the organization and expand its business. The CEO of FDS decided to internally create positions and space for the new roles instead of hiring from outside. This seemed to be a prudent decision because the demands of a trainer role, as an expert in FDS products, can be best filled by a consultant. It is also the most cost effective: as trainer, a consultant already has knowledge of FDS products and exposure to knowledge transfer sessions with customers.

- **PRE_BS_4.** “Assistance of third party experts can be used to help develop the AC Academy training curriculum.” (Table 8-7: Item 4) → **EARLY_BS_1:** “FDS Academy would like to engage services of a consulting company for the LMS selection process. A RFP was sent out with a list of functional requirements. See Table 8-3. Requirements derived from RFP.” (Table 8-7: Item 9)

In two occasions, the FDS Academy sought the expertise of external consultants in realizing two key training artifacts: a training curriculum and a learning infrastructure. Utilizing external consultants was the means sought by the FDS Academy to overcome its lack of training expertise and to quickly realize a training curriculum. With a positive experience in involving consultants for a project, the FDS Academy once again sought the expertise of another consulting company for the LMS project. This way, the two statements are related because the former action sets precedence for the next. In other words, the choice to once again involve consultants for another project is plausible because it has already been done before.

- **EARLY_BS_3.** “FDS Academy would like to sign Oracle iLearning hosting contract.” (Table 8-7: Item 11) → **EARLY_BS_14:** “The trainer should promote the use of Oracle iLearning in training sessions.” (Table 8-7: Item 22)

These two statements are an example of solution reinforcement: it follows that in deciding to acquire to support training and eventually classroom-based product training, the trainer must be encouraged to make use of it. The latter statement came about because according to the FDS Manager, the trainer tends to forget about the availability of the lecture slides in the LMS. This happens because the trainer is not used to having the system before and is not involved in administering content with the LMS. Therefore, such agreements in the team have to be made to ensure that Oracle iLearning gets promoted.

- **EARLY_BS_6.** “Handouts and lecture slides should not be stored in the Oracle iLearning online server. They should be stored on a separate local server which interfaces with Oracle iLearning.” (Table 8-7: Item 14) → **EARLY_BS_7.** “Updating existing slides and materials should be done via the local training server.” (Table 8-7: Item 15)

In the same way, these two business statements are comparable to the two previous statements. The difference however is that **EARLY_BS_6** reinforces the decision stated in **EARLY_BS_5** as a form of action instead of a policy of an agreement. Specifically, **EARLY_BS_6** is an emergent procedure specification that immediately results from deciding to have a separate physical location to store the training slides. It is an attempt to work smarter.
Altogether, these four instances of related business solution statements indicate that there is more
to within domain evolution than a simple change in the set of business solution statements in the
domain over time. Within the domain, business solution statements also seem to impact each
other by way supporting a business decision with another business decision, policy or informal
agreements and specifications of new procedures.

8.5.3 Software product concept domain evolution

The evolution of the software product concept domain in terms of requirements statements is
shown in Table 8-8. Out of this compilation, we can trace the progression of mental models
regarding training and ICT support for training. Specifically, we can draw the following pattern
of concepts prevailing in the implementation: classical training → e-learning → Learning
Management System (LMS) idea → Oracle iLearning → blended learning → Oracle: other uses
→ moving away from Oracle iLearning → open source LMS → Oracle iLearning abandonment.

The pattern of concepts more or less tells the story of the LMS implementation as it unfolded. Starting
from pre-implementation with training as its key functional role, the FDS Academy has
worked out the concept of training in its classical form: classroom-based, instructor-led. Out of
this mental model of its key task, the FDS Academy turned its attention to e-learning as a way of
going with the latest trend and at the same time to find ways on how to implement an efficient
training approach to inducting new employees. Only after realizing that having e-learning
modules necessitate an infrastructure for their delivery, access and maintenance as training
materials did the concept of a learning management system (LMS) become understood. From the
idea of a learning management system, a specific product in the form of Oracle iLearning was
discovered. Consequently, having an LMS in place expands the possibilities for training and the
team opportunistically took advantage and made use of the concept of blended-learning. Next to
blended-learning, the Oracle iLearning was also quickly introduced to other purposes such as
sales and account management: prospective clients can have temporary access to the e-
curriculum. Oracle iLearning also proved to be an effective customer management tool,
especially during times when a customer cannot wait for training and insists on a date that is not
feasible for the team. To work out a win-win situation and to avoid issues with the customer, the
e-learning curriculum is offered to the customer as an interim solution for a blended-learning
training session that will be set at a mutually agreed schedule. However, as system performance
deteriorates, there was a noticeable shift in interest to other software applications that can replace
the problematic functionalities in Oracle iLearning. When the team became part of the beta-
testing group as a result of the buy-out, the disposition to move away from Oracle iLearning
became stronger as the priorities and interests of the new manager are clearly different. The
preference is for open source LMS systems. Finally, when the financial crisis became a grave
phenomenon to be dealt with, mental models of cost reduction and termination became
commonplace. For Oracle iLearning, these mental models hold.

It is interesting to note that a comparatively similar pattern can also be found in the other two
cases of learning or course management system implementations in this study. The similarity is in
how the implementing teams came to the concept of learning infrastructures and eventually of
products such as Oracle iLearning, TeleTOP (Case C) and First Class (Case B). Basically, it all
begins with the vague idea of wanting to innovate, or in this case as the FDS Academy manager
mentioned, going with the trend. The first innovation concept that they come up which is related
to training and education is e-learning. Apparently, the concept of e-learning is only vaguely
understood because it comes in many forms. It can be a simple stand-alone computer based
instruction or a series of hand-outs and slides made available through the web. It makes a
difference also if the instruction is individualized or given to groups. Oftentimes, it is associated
with distance learning, but even that is also a not well-defined instructional arrangement. In any
case, when these implementing teams try to further work out what e-learning means for them, they come to logistics and administrative aspects of the concept. In other words, they grapple with the how of e-learning and less of its what. In doing so, they get to learn about learning or course management systems and they decide which specific product is most suitable for them. The choice for Oracle iLearning in this case is a good example. Finally, with the system in place, other ideas get generated, either opportunistic or emergent in about to further make use of the LMS.

Table 8-8. Software solution specification statements: all phases

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_SP_1</td>
<td>FDS Academy, tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SP_2</td>
<td>Oracle iLearning proposed as affordable and suitable LMS alternative; FDS Academy should try out this product.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SP_3</td>
<td>Oracle iLearning will be the default environment for delivering training in e-learning format; this is applicable especially to new employee training.</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning.</td>
</tr>
<tr>
<td>5</td>
<td>POST_SP_1</td>
<td>Oracle iLearning should be seen as a sales and account management support tool.</td>
</tr>
<tr>
<td>6</td>
<td>POST_SP_2</td>
<td>Course evaluation doesn’t have to take place in Oracle iLearning anymore.</td>
</tr>
<tr>
<td>7</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>8</td>
<td>POST_SP_4</td>
<td>With the dissolution of the training department, Oracle iLearning is no longer needed. Subscription has to be at minimum to meet contractual obligation with customers.</td>
</tr>
<tr>
<td>9</td>
<td>POST_SP_5</td>
<td>Other e-learning authoring tool, i.e. Adobe Captivate should be used for creating assessments instead of Oracle iLearning.</td>
</tr>
<tr>
<td>10</td>
<td>POST_SP_6</td>
<td>[OpenSrcLMS] Course delivery: Training portal idea?</td>
</tr>
<tr>
<td>11</td>
<td>POST_SP_7</td>
<td>[OpenSrcLMS] Technological: What to do with Dickens? meaning what is the implication to the existing training environment separate from Oracle iLearning.</td>
</tr>
</tbody>
</table>

To wrap up this discussion on software product concept domain evolution, we would like to identify 3 indicators of drift captured by the software product concept statements.

- EARLY_SP_3. “Oracle iLearning will be the default environment for delivering training in e-learning format; this is applicable especially to new employee training.” (Table 8-8: Item 3) → EARLY_SP_4: “As e-learning software, Oracle iLearning LMS will support blended learning.” (Table 8-8: Item 4)

These two statements represent the shift in the idea of how training should be carried out and how Oracle iLearning should fit into those formats. In the first pre-implementation, Oracle iLearning was meant to introduce the concept of e-learning to the organization and support the induction of new employees. The view on e-learning quickly shifted into blended when classroom-based product training is upgraded to include e-learning support from the LMS.

- EARLY_SP_2. “Oracle iLearning proposed as affordable and suitable LMS alternative; FDS Academy should try out this product.” (Table 8-8, #2) → POST_SP_3: “Oracle iLearning should cost less and must be replaced by an open source LMS.” (Table 8-8: Item 7)

At the later part of implementation, there is a shift in interest from a subscription-based hosted service towards an open source LMS.

- EARLY_SP_4. “As e-learning software, Oracle iLearning LMS will support blended
learning.” (Table 8-8, #4) → POST_SP_6: “Course delivery: Training portal idea?” (Table 8-8: Item 10)

Based on the how the system is being used to manage the delivery of training, other ideas come about such as a training portal use for Oracle iLearning. While a training portal concept is also a vague idea, the intention is to channel the system into a more publicly accessible page, a portal in this sense, where on-going training sessions are made visible and trainees can self-enroll. Therefore, the drift is from a more restrictive system towards an open access.

8.5.4 Software solution specification domain evolution

We compiled the software solution specification statements in two tables: Tables 8-9 and 8-10. Table 8-9 is a listing of the statements gathered from the case without the written specifications based on the RFP or the open source LMS project. Table 8-10 on the other hand is a compilation of the RFP and open source LMS project requirements.

Table 8-9. Software solution specification statements compilation without RFP and open source LMS statements

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_SS_F_1</td>
<td>FDS should have a trial account to try out Oracle iLearning.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SS_Q_1</td>
<td>Staff should receive training in Oracle iLearning.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SS_Q_2</td>
<td>Oracle iLearning LMS site must conform to FDS look and feel</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SS_F_3</td>
<td>Oracle iLearning should support multiple curricula, i.e. it should be possible to incorporate Powerpoint slides and exercises in pdf format used in classroom-based training.</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SS_F_4</td>
<td>Oracle iLearning subscription is limited to 100 users. For each user, FDS is entitled to 5 MB of space.</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_F_5</td>
<td>Hand-outs and lecture slides should be available in Oracle iLearning only for the duration of the training session. Trainee accounts should expire at the end of the training session.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_SS_F_6</td>
<td>It should be possible within Oracle iLearning to link to materials in external servers and display these correctly.</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_SS_F_7</td>
<td>Employee accounts do not expire.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_SS_F_8</td>
<td>The Oracle iLearning site for trainees at customer site will not contain powerpoint slides; however, they can view the training content outline.</td>
</tr>
<tr>
<td>10</td>
<td>EARLY_SS_F_9</td>
<td>Course evaluation should be done via Oracle iLearning. Trainees should be able to fill-in an online course evaluation form.</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_F_10</td>
<td>Reporting in Oracle iLearning requires knowledge of databases. Results of course evaluation are in raw format requiring transformation into other formats distributable per email.</td>
</tr>
<tr>
<td>12</td>
<td>EARLY_SS_F_11</td>
<td>Two new courses are added in the FDS curriculum. This should also be incorporated in the Oracle FDS iLearning site.</td>
</tr>
<tr>
<td>13</td>
<td>POST_SS_F_1</td>
<td>User subscription should be increased to 135 users.</td>
</tr>
<tr>
<td>14</td>
<td>POST_SS_Q_1</td>
<td>Training materials, including the FDS Oracle iLearning site should be consistent with the new housestyle.</td>
</tr>
<tr>
<td>15</td>
<td>POST_SS_F_2</td>
<td>Oracle iLearning subscription should be reduced to 100 users.</td>
</tr>
<tr>
<td>16</td>
<td>POST_SS_F_3</td>
<td>Employee accounts should expire. Accounts of those who have already undergone new employee training should be deleted.</td>
</tr>
</tbody>
</table>

Just as in the previous cases, the software solution specification domain is also the most represented requirements domain in this study. This is further accentuated by the availability of written requirements for the same software product concept across different periods. A particular feature of this implementation is the engagement of external parties, i.e. consultants and interns, in writing the specifications. On the basis of these written specifications (Table 8-10), we found
out that the requirements did not really change. For example, the functionality for supporting Powerpoint slides and text-based documents EARLY_SS_F_14 (Table 8-10: Item 8) reported in the RFP is the same specifications mentioned in POST_SS_F_4 (Table 8-10: Item 25) and POST_SS_F_5 (Table 8-10: Item 26) given in the open source LMS requirements. The same holds true for EARLY_SS_F_17 and POST_SS_F_8 to POST_SS_F_18 (Table 8-10: Items 12 and 29 through 39), only that the latter contains more detail about assessment types. We can say that since Oracle iLearning is the resulting product out of the RFP, it more or less meets the early requirements. However, since requirements did not really change except for the specific desire for an open source system in post deployment, Oracle iLearning still satisfies most of the requirements, only that it is not open source software.

However, beyond the similarities in the written specifications, there are also observable associations among the specifications which sustain domain evolution. We can say that one specification is associated with another in a way that this refines it by adding more details or expanding its scope. In other words, a specification statement is clarified and updated one or more specification statements from the same domain. It is an impact relation of mechanism but taking place within the same domain. We have identified the following instances of software solution specification refinements and updates in this case. For purposes of brevity, we will make use of the codes instead of statements. For the corresponding statements, please refer to Tables 8-9 and 8-10. What each series of codes refer to is implied in the discussion of each example.

- **EARLY_SS_F_3 → POST_SS_F_1 → POST_SS_F_2 → POST_SS_F_26**
  (Table 8-9: Item 4 → Table 8-9: Item 13 → Table 8-9: Item 15 → Table 8-10: Item 47)

  This string of specification statements refers to the desired number of users that will be supported by the system. The first three specifically refer to Oracle iLearning while the last one is a specification for an open source LMS. In this series of requirements, the specifications align with the organizational need and context. From a given limit of 100 users (EARLY_SS_F_3), the specification was adjusted to 135 users (POST_SS_F_1). However, as a result of re-organization and changing priorities, the number of subscribed users is reduced to 100. On the other hand, pushing Oracle iLearning aside, it would appear that for an open source LMS, the ideal number of users to be supported is 200 (POST_SS_F_28). The specification on the desired number of users, as it came along, is a pattern of fluctuating values.

- **EARLY_SS_F_7 → POST_SS_F_3** (Table 8-9: Item 8 → Table 8-9: Item 16)

  This is a form of specification update in which the latter reverses or invalidates the former. In these two specifications, the property of non expiring employee accounts stated in early implementation is reversed to expiring accounts in post deployment.

- **EARLY_SS_F_17 → POST_SS_F_8 → POST_SS_F_18**
  (Table 8-10: Item 12 → Table 8-10: Items 29 through 39)

  Assessment utility for creating scored exams is one of the functional specifications written in the RFP back when FDS Academy was in search for a system. This is updated in the later period when the team started to consider option source options. What type of exams, grading system, administration and other specific details of implementing assessment as system functionality are specified in the open source LMS project requirements.

- **EARLY_SS_F_3 → POST_SS_F_19 & POST_SS_F_20**
  (Table 8-9: Item 4 → Table 8-10: Items 19 and 20)

  Next to using Oracle iLearning for new employee training and hosting e-learning modules, one of the requirements that grew in the early period is the support needed for blended learning.
This means that the new employee training modules and the classroom-based training materials should co-exist in the LMS. At some later period, with new courses being added to the system, the need to create curricula on the fly from existing ones became a requirement (POST_SS_F_19). At the same time, it was also known to the group it was not possible in Oracle iLearning to run two curricula in parallel while there was a need to do so. This is a requirement discovered later on, which together with the previous one updates the original requirement for supporting multiple curricula (EARLY_SS_F_3).

Table 8-10. Software solution specification statements compilation: RFP and open source LMS requirements

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EARLY_SS_Q_3</td>
<td>[RFP1] A hosted-solution, so that the company is not required to tax internal IT infrastructures to support the roll-out of the FDS Academy.</td>
</tr>
<tr>
<td>2</td>
<td>EARLY_SS_Q_4</td>
<td>[RFP2] Option of potentially relocating the solution to an internal web server in the future as demand increases.</td>
</tr>
<tr>
<td>3</td>
<td>EARLY_SS_F_12</td>
<td>[RFP3] Prefer an SQL database format (to be compliant with internal systems).</td>
</tr>
<tr>
<td>4</td>
<td>EARLY_SS_F_13</td>
<td>[RFP4] The company will give strong consideration to those vendors that have built-in authoring capability or can recommend a seamless content authoring approach. Asset Control is additionally considering the use of Trivantis, Lectora Publisher for content authoring and is still seeking a solution for rapid creation of software application simulations (i.e. RapidBuilder from XStream Software or OnDemand from Global Knowledge).</td>
</tr>
<tr>
<td>5</td>
<td>EARLY_SS_F_14</td>
<td>[RFP5] Performance tracking for asynchronous course material (using industry standards).</td>
</tr>
<tr>
<td>6</td>
<td>EARLY_SS_Q_5</td>
<td>[RFP6] A strong security module, allowing for appropriate access to course content and administrative data.</td>
</tr>
<tr>
<td>7</td>
<td>EARLY_SS_F_15</td>
<td>[RFP7] Threaded Discussion Groups (preferably linked to specific courses).</td>
</tr>
<tr>
<td>8</td>
<td>EARLY_SS_F_16</td>
<td>[RFP8] Ability to attach PowerPoint and Word Documents as reference material.</td>
</tr>
<tr>
<td>9</td>
<td>EARLY_SS_F_17</td>
<td>[RFP9] Integrated email collaboration (learner to instructor, learner-to-learner, etc.).</td>
</tr>
<tr>
<td>11</td>
<td>EARLY_SS_Q_6</td>
<td>[RFP11] 24X7 support.</td>
</tr>
<tr>
<td>13</td>
<td>EARLY_SS_F_20</td>
<td>[RFP13] Level-1 Evaluation Utility (creating “happy sheets”).</td>
</tr>
<tr>
<td>14</td>
<td>EARLY_SS_Q_7</td>
<td>[RFP14] Regulatory Compliance.</td>
</tr>
<tr>
<td>16</td>
<td>EARLY_SS_F_22</td>
<td>[RFP16] Doesn’t require a plug-in.</td>
</tr>
<tr>
<td>17</td>
<td>EARLY_SS_F_23</td>
<td>[RFP17] Classroom management (may use in the future, but not part of initial implementation).</td>
</tr>
<tr>
<td>19</td>
<td>EARLY_SS_F_25</td>
<td>[RFP19] Competency Management (future).</td>
</tr>
<tr>
<td>20</td>
<td>EARLY_SS_F_26</td>
<td>[RFP20] 360-degree evaluation (preferred for future).</td>
</tr>
<tr>
<td>22</td>
<td>EARLY_SS_Q_8</td>
<td>[RFP22] Even though FDS is located in the Netherlands, all training sessions will be conducted in English. There is no need for multi-lingual support or localization.</td>
</tr>
<tr>
<td>23</td>
<td>EARLY_SS_Q_9</td>
<td>[RFP23] The company has no immediate plans for providing e-commerce support (credit card transactions) in the initial implementation.</td>
</tr>
<tr>
<td>24</td>
<td>EARLY_SS_Q_10</td>
<td>[RFP24] There is no requirement at this time to integrate performance data with an external ERP or CRM system. This should be a fairly straightforward implementation of your hosted-LMS offering.</td>
</tr>
<tr>
<td>26</td>
<td>POST_SS_F_5</td>
<td>[OpenSrlSMS] Course delivery: PDF’s usable (nice to have).</td>
</tr>
<tr>
<td>27</td>
<td>POST_SS_F_6</td>
<td>[OpenSrlSMS] Course delivery: Place to store assignments and reading material per lesson/module.</td>
</tr>
<tr>
<td>28</td>
<td>POST_SS_F_7</td>
<td>[OpenSrlSMS] Course delivery: Course calendar, preferably smoothly looking.</td>
</tr>
<tr>
<td>31</td>
<td>POST_SS_F_10</td>
<td>[OpenSrlSMS] Course delivery 4Assessments: Types of test: With pictures?</td>
</tr>
<tr>
<td>33</td>
<td>POST_SS_F_12</td>
<td>[OpenSrlSMS] Course delivery 4Assessments: Display scores and transcripts.</td>
</tr>
</tbody>
</table>
8.6 Impact relations

Based on already known impact relations (M1-M7), we identify instances of inter-domain requirements evolution that appropriately depict each change mechanism. For each impact relation, the relevant requirements statements are presented in table format. For presentation purposes, the impact relation tables are provided at the end of this chapter in Appendix C through I.

8.6.1 M1. Business problem resolved by business solution

The resolution of business problems by business solutions indicating requirements change is a regular feature of this groupware implementation. This can be found in the list of M1 impact relation instances given in Appendix C: Case D. Notable in these M1 instances are the two levels of resolution, which were also observed in the other cases. Firstly, there is the resolution of
business problems by business solutions without the need for or resorting to software. Secondly, when the relevant business problem ultimately leads to system implementation, the business problems associated with the implementation also find resolution through business solutions. In all these M1 instances, the overriding trigger for change is a breakdown in the form of inability to meet customer demands, the lack of capacity to deliver new services, a looming economic crisis, etc.

For the first type of M1 resolutions, we can see examples of business problem and business solution statements as impact relations not bound by software (Appendix C: Case D, Items 19 through 27). These impact relations representing the pre-implementation phase of the implementation don’t particularly indicate a global software solution, although they provide the context for the eventual implementation of Oracle iLearning. Specifically, the problem statements identify the breakdowns the organization is facing: increasing demands from customers to make the software product more accessible and the need to expand organizational capacity in order to be able to offer additional services such as training. Having the problems known this way, we can say that the organization was aware of and understood the challenges it was facing. Accordingly, it was able to come up with solutions that in a way appropriately addressed some of the problems. For example, the creation of a training department is warranted (Appendix C: Case D, Item 19) and needed. Likewise, the combination of solutions that focus first into optimizing internal resources (Appendix C: Case D, Item 20) and seeking expert help outside the organization is a way of overcoming resource and knowledge constraints.

With the implementation of Oracle iLearning, new sets of problem arise as the organizational situation has also changed. There is for example the increasing demand for onsite training requested not according to protocol and the presence of the system contributed partly in resolving the issue (Appendix C: Case D, Item 28). There is also the instance when the implemented software negatively influences the quality of training (Appendix C: Case D, Item 30) and this was resolved by resorting to the use of other software. However, while we do talk of software in this instance, the business solution is not about improving the software in question but rather about using other software. More specifically, it is about a task carried out with use of other software. Finally, there is the overwhelming problem of financial viability that is threatening the organization of which an expected action is to cut down cost. We notice that in these business problem and business solution resolution statements in the post-deployment phase problems brought about or influenced by the software implementation find resolution in business solutions that do not specify software functionality, although the solution entails the use of software.

8.6.2 M2. Business solution supported by a software product concept

Appendix D lists the requirements statements depicting the resolution of business solutions by extending these into software ideas. The first business solution statement amenable to a software solution that enters our list is the definition of training as service delivered to FDS customers (Appendix D: Case D, Item 14). Training as classroom-based and instructor-led is very much amenable to ICT support in the light of innovations in e-learning trends and its growing popularity which the FDS Academy was aware of. Therefore, it is not a surprise that the classical form of training got upgraded to blended learning which incorporates e-learning, when the appropriate software gets implemented (Appendix D: Case D, Items 15 and 18).

Likewise, following the suggestion of their consultant to try out Oracle iLearning leads to acquiring the system (Appendix D: Case D, Item 16). Investing time and effort to try a candidate software provides the experience to get to know the product. They help in formulating a mental model that favorably views the software appropriate, specifically to the priority placed on a short induction period for new employees.

In the evolving use of Oracle iLearning, we have also encountered a shift in product concept
which promotes the use of the LMS as a sales and account management tool. This is in relation to
the stopgap measure taken to satisfy customer needs (Appendix D: Case D, Item 19). Ultimately,
this promoted an improvised use of Oracle iLearning.

In the later phases of implementation, there is an observable tendency within the team to seek
solutions outside Oracle iLearning in order to overcome issues relating to its use. In most
instances, the team sought solution in terms of other software. We listed the business solution
statement POST BS 3 (Appendix D: Case D, Item 20) as an example of this, in which the
training coordinator opportunistically made use of the Marketing Department’s online survey
system to create a new course evaluation form. This action creates the perception that course
evaluation, which is an integral part of any training process, doesn’t have to take place in Oracle
iLearning. Such perceptions actually diminish the popularity of the system to the team.

Lastly, drastic measures such as the dissolution of the training team in order to stay
economically feasible also meant that associated resources that entail cost will also become
obsolete or not needed. This way of handling the breakdown translates to the non-renewal and
eventual abandonment of Oracle iLearning (Appendix D: Case D, item 21)

8.6.3 M3. Software product concept resolved by software solution specification

Various instances of software product concept resolutions through software solution
specifications found in this case are enumerated in Appendix E. The list shows (Case D) that this
type of impact relation is motivated by breakdowns and initiatives.

The first instance in the list, found from the early implementation period is about the FDS
Academy’s view of ICT applications for training and how this translates into specific software
specifications (Appendix E: Case D, Item 28). As discussed previously, the FDS Academy is
faced with the problem of how to reduce the time needed to train new employees. It believes that
e-learning solutions can help in solving this problem. Knowing this, the team was able to come
up with a more specific formulation of its solution specification, namely that of a hosted solution
so that this will not tax FDS internal IT resources (Appendix E: Case D, Item 28). However, the
option to migrate the solution in-house at some time in the future remains (Appendix E: Case D,
Item 29). This basically means that the desired solution should be flexible in terms of where it
will be hosted. Finally, while not explicitly stated as a requirement, the non-expiration of
employee accounts in the resulting LMS implementation is a consequence of associating the e-
learning solution with new employee training (Appendix E: Case D, Item 30).

When the RFP process failed to deliver an LMS product, an alternative product, Oracle
iLearning, was proposed. When it was proposed, the specifications of what it can do, what are its
attributes except for the fact that is a web-based hosted LMS known to the consultants are not
well-known to the FDS Academy. At that time, Oracle iLearning was a vague image of a software
product representing an LMS. In order to be able to try it out, it should be possible for a hosted
Oracle iLearning site to allow trial accounts (Appendix E: Case D, Item 31). Being able to realize
this, such facilitation of a trial account eventually led to the signing of a hosting agreement
between FDS Academy and Oracle iLearning. In order to realize the e-learning and training
delivery responsibilities assigned to Oracle iLearning, several specifications have to be realized
such as training in Oracle (Appendix E: Case D, Item 32) Naturally, as an application
representing a service, it should conform to the company’s look and feel (Appendix E: Case D,
item 33). In this instance the software product concept and software solution specification do not
to be related to each other sequentially. The software specifications were written long after the
software product concept is known. In this impact relation (Appendix E: Case D, Item 33) the
specification about implementing the company’s look and feel to the learning is not a direct result
of having Oracle iLearning associated with the e-learning concept. Rather, the specification is a
later consequence of having an e-learning infrastructure and using it.
The software product concept EARLY_SP_4 (Appendix E: Case D, Item 34) captures the expansion in use of Oracle iLearning from new employee training into blended learning initiatives for its flagship product training services. This expansion in concept implies that Oracle iLearning should have appropriate functional properties that enable blended learning. Among these include (i) support for multiple curricula, i.e. it should be possible to host both product training curriculum and new employee training curriculum in one hosted LMS (Appendix E: Case D, item 34), (ii) temporal availability of and access to training slides accompanying a training session as well as course attendees (Appendix E: Case D, Item 35), and (iii) support for stand-alone, external third-party e-learning authoring tools such as Lectora (Appendix E: Case D, Item 36). In these instances, the resulting software solution specifications provide justification to the concept of blended learning as way of resolving the software product concept.

As the interest in Oracle iLearning wanes motivated by cost reduction initiatives lead to two specific software design decisions (Appendix E: Case D, Items 38 and 39). First is the capability to reduce subscribed users in the middle of the year (Appendix E: Case D, item 37). The second is the specification for a new LMS application which is completely open source (Appendix E: Case D, Item 38).

Finally, the mindset that pushes for the use of authoring tools in creating assessments is met with a corresponding specification for an open source LMS that requires import capability with those tools (Appendix E: Case D, Item 39).

8.6.4 M4. Software solution specification supported by business solution

This case provides a number of interesting instances of M4 impact relations which illustrate the various ways of how Oracle iLearning implementation was supported and stimulated. They represent creative ways of how the team managed to overcome breakdowns in Oracle iLearning us and adjusted its way of working to accommodate the system. These conscious decisions and actions helped in sustaining the continued implementation of Oracle iLearning.

The compilation of M4 impact relations is given in Appendix F, which also shows that impact relations of this type found in this case are largely breakdown-driven. Providing the context for is the failed RFP process which prompted the external consultants to propose an alternative solution (Appendix F: Case D, Item 12). The alternative solution is a COTS web-based LMS Oracle iLearning. To realize this as a solution, it should be possible for third parties to try it out. Having this property enabled the FDS Academy to make use of it as a demo environment using real training data. Such effort taken by team was a step towards the desired direction because they can build and proceed from the demo environment. Therefore, with a visible demo training environment, the decision to sign up Oracle iLearning as the default LMS for the organization was simplified and made easier (Appendix F: Case D, items 13-14).

Some of the terms of the hosting agreement restrict the functional use of Oracle iLearning, thereby leading to breakdowns. The first one of this is the limit of 100 subscribers which translate to a limit in server space. There isn’t enough space to host the classroom-based product training materials. Therefore, the team made the decision of not uploading the training slides to the hosting server and instead had put up an internal in-house server to host training materials (Appendix F: Case D, Item 15). A consequence of this decision to overcome the same breakdown is a new of working: updating training materials should be done in the local in-house server instead of in Oracle iLearning (Appendix F: Case D, Item 16). This means that whenever a training session will be setup for an upcoming training, the configuration will be done in Oracle iLearning whereas the update on the raw materials will be done on the local training server. Effort must also be expended to ensure that the linkage between Oracle iLearning and the local training server works properly.

The software specification EARLY_SS_F_8 (Appendix F: Case D, Item 17) was also a
consequence of conserving the hosting space and implementing an in-house training server to host the training materials. It is a breakdown experienced by onsite training participants. They cannot view the accompanying Powerpoint slides of an ongoing lecture because from their location Oracle iLearning cannot link to the local training server. As a solution, trainees were provided with USB sticks containing the training slides. However, access to the training materials in the USB sticks is manual and independent of Oracle iLearning. Access to the training materials in the USB sticks is not integrated with Oracle iLearning.

Another effort to make the system work to meet the needs of training operations is captured in the impact relation EARLY_SS_F_9 → EARLY_BS_9 (Appendix F: Case D, Item 18). This is about the need to have the course evaluation online and preferably hosted in the LMS. Strictly speaking, the system does not have an explicit functionality for creating course evaluations. However, it has a module for creating assessments and this can be used to create course evaluation. Therefore, the need for online course evaluation form is realized by using the assessment module of Oracle iLearning.

Processing the results of the filled-in online course evaluation forms is another matter and issue than creating the forms in the system. The team discovered that in order to make the results of the course evaluation communicable and interpretable, another way of working with data has to be done. The reporting functionality in Oracle iLearning does not lend the desired functionality because the inputs are stored in a database and in order to manipulate this data, knowledge of database is needed. Unfortunately, for the team, the knowledge and the time to look into this was limited, if not absent. Therefore another approach was devised and that is to use MS Excel to process the raw data. That is by first capturing the data shown on the course evaluation page and transferring it to MS Excel through copy and paste method (Appendix F: Case D, Item 19).

The course evaluation in Oracle iLearning, while it fulfills the team’s training requirements, has always been a source of breakdowns. In the previous paragraph, we discussed the issue with the transformation and communication of the course evaluation results. The next impact relation instance in our list refers to the same course evaluation functionality in Oracle iLearning but whose performance over time has deteriorated such that in the period that follows, it was migrated into another environment and system (Appendix F: Case D, Item 20).

Finally, in item #21 (Appendix F: Case D) of our M4 impact relations list, we also added the connection between the supported specification for integration and compatibility with e-learning authoring tools (EARLY_SS_F_13) and the resolution to use Oracle iLearning hosting the standalone e-learning modules as an interim onsite training solution (POST_BS_2). As the list shows, this impact relation spans from early implementation to post-deployment. This means that the update to the earlier requirement took place at later time when the problem of urgent onsite training requests arose. Obviously, support for stand-alone e-learning modules was possible in Oracle iLearning given that it was used for new employee training. The combination of available suitable content – the e-learning modules, and the availability of Oracle iLearning as an online infrastructure and medium enabled the team to come up with an acceptable compromise with customers by providing them access to training materials while negotiating for an onsite training session that will come later. It was then a win-win situation.

8.6.5 M5. Business solutions leading to improved software solution specifications

Various instances of M5 impact relations (Appendix G: Case D) in this case show how business solutions enacted through decisions and actions by the implementing team help in improving software specifications. Through such efforts, the team was able to properly identify its needs and move a step closer towards an ideal system.

The decision to employ an external consultant to assist in the selection and evaluation of suitable ICT applications to support training for new employees is one good example of a
business solution that motives an M5 impact relation. This tactic led to the formulation of an RFP that outlines concrete specifications for a learning management system (Appendix G: Case D, Item 18). This action is once again repeated in the post-deployment period in which the prevailing choice for an open source LMS was supported by the idea of using an intern to gather the team’s requirements for a new open source LMS (Appendix G: Case D, Item 27). Both actions can be regarded as initiatives.

The outcome of the RFP process is the choice for a specific LMS application. The FDS Academy decided to sign up Oracle iLearning as LMS of choice and as platform for new employee training. This action led to the formulation of specific requirements for Oracle iLearning such as training (Appendix G: Case D, Item 19) and conformance with the company’s look and feel (Appendix G: Case D, Item 20).

In an attempt to improve the quality of new employee training, the team also embarked on including technical training in the program. Specifically, it planned on developing stand-alone modules for FDS product training to be incorporated during the orientation session. As previously discussed, the general idea to have a half-day orientation session and a half-day e-learning session for which no speakers were required. The implication of this requirement is that Oracle iLearning must support the integration of stand-alone e-learning modules developed in other e-learning authoring tools, such as Rapid Builder and Lectora. Apparently, this requirement was already known and captured in the RFP (Appendix G: Case D, Item 21). Consequently, using the system to for address the need for training new employees had brought about an implicit requirement that employee accounts in Oracle iLearning should not expire (Appendix G: Case D, Item 22).

After modifying or finalizing the training program for new employee training, the focus turned to the classroom-based, instructor-led product training. Using the LMS to support training delivery was a learning process that brought up the concept of blended-learning (Appendix G: Case D, EARLY_BS_5). The expansion and re-definition of product training into blended learning entails that Oracle iLearning must (i) support multiple curricula (Appendix G: Case D, Item 23) (ii) support the Powerpoint slides and Word documents (Appendix G: Case D, Item 24) and (iii) ensure the availability of relevant training materials online should coincide with the duration of the training session (Appendix G: Case D, Item 25). Needless to say, the requirements for supporting blended-learning were already specified before the concept of blended-learning was conceived.

The limitations imposed by the hosting agreement on the availability of server space prompted the team to find another solution for the need to put up training materials and handouts online. A solution was found by adjusting work processes and utilizing another resource (Appendix G: Case D, item 26). This is by acquiring a local in-house server to host the training materials instead of uploading these into Oracle iLearning directly. This means that for Oracle iLearning, it should be possible to link the web-based course and its training materials to a local in-house server and display these properly (Appendix F: Case D, item 26). The desired result is that the learners should not be able to notice the transition from a central host to an in-house server while accessing various components of the training.

Items 28 through 39 (Appendix F: Case D) are M5 impact relations that illustrate how desired improvements to training products can steer the formulation of software specifications and the elaboration of a needed functionality. Specifically in the post-deployment phase, sufficient attention was given to specifying the needed Assessment functionality in the open source LMS project requirements.

8.6.6 M6. Software solution specifications leading to business problems

It became remarkably apparent in the later phases of implementation that the system has been
causing problems. This mechanism is known to our framework as the M6 impact relation: Software solutions leading to new business problems. Most prominent of these breakdowns is the increasing deterioration in performance and the quality of access to learning materials (Appendix H: Case D, item 19). Eventually, these performance breakdowns led to deterioration in the quality of training reinforced by negative trainee feedbacks (Appendix H: Case D, Items 20 and 21). These breakdowns are induced by the software itself as a whole and a few of its specific properties, i.e. support for multiple curricula and attachments such as Powerpoint slides and Word documents (Appendix H: Case D, Items 20 and 21).

It was also known in this case that the specifications for a desired LMS already created problems even before it got implemented. The suppliers who responded to the RFP were too expensive for the manager, therefore it was not acceptable (Appendix H: Case D, Item 18).

The other impact relations instances in this case show how initiatives turn into breakdowns because the existing software product property is bogged down by poor interface usability. For example, Oracle iLearning remains a difficult system to use for some members of the team (Appendix H: Case D, Item 22). There are many roles within the system but no one in the team knows what all these roles are and what kind of rights they have in the system. There are a lot of functions, buttons and links whose representations do not clearly indicate what their purpose is. For one particular member, the system is too complicated to use. She cannot reproduce on her own the steps taken to carry out a task in Oracle iLearning. Specifically, this applies to creating assessments (Appendix H: Case D, Item 23). The common complaint is that tutorials and help on how to create assessments seem to be unavailable (Appendix H: Case D, Item 24).

Finally, that the costs associated with subscribing to an LMS application were being regulated by the number of subscribed users was a problem for the new manager who was interested in cutting down costs (Appendix H: Case D, Item 25).

8.6.7 M7. Business problems resolved by software product concepts

There are many instances in which while trying to structure and analyze a problem, a solution in the form of software is thought of and introduced. Motivated usually by breakdowns in this case, the instances of M7 impact relations that we found are detailed in Appendix I.

The very first association with software as a general solution to a prevailing business problem was found between PRE_BP_6 and EARLY_SP_1 (Appendix I: Case D). While not very specific yet as to what application class it should look into, the FDS Academy sought ICT solutions that can help in addressing the urgent need to train and equip new employees with knowledge and skills. The concept of ICT and e-learning and learning management systems came a little bit later. Basically, what the team was interested in was in finding out what the trends were.

Secondly, a failed RFP process is problematic not only for customer organization but also for the consultant. As a process intended to solicit the appropriate solution, the failure of the RFP process was repaired through the suggestion of a specific software product in the form of Oracle iLearning (Appendix I: Case D).

The popularity of training as a service to FDS customers brought along problems with it. Other departments interfacing directly with customers quickly come to the understanding that training can address knowledge gaps on FDS solutions. Therefore, the demand for onsite training continued to grow and while not the target of FDS training, requests were also being made for prospects. While this issue can be addressed through policies and agreements which can be escalated to higher management in order to avoid further disputes, the team however sought to address the insistent training demands by offering an alternative interim solution, that is by enabling access to the e-learning modules hosted in the LMS system. Therefore, the use of Oracle iLearning was already shifting to an account management and sales tool (Appendix I: Case D,
8.6 Impact relations

Item 14).

The organizational restructuring due to the buy out which resulted into a new team structure for the FDS Academy (which became FDS Training without its original manager) was a difficult process for the team. It was appended to the beta-testing group whose manager became the new training manager. There were conflicts between the old team and the new manager ranging from knowledge of training operations with each department having their own sub-culture, consolidation in one location and differing business priorities. To begin with, the original training team is a profitable department within FDS whereas the beta-testing group is a non-revenue generating. Ultimately, the issue boiled down to business priorities in which the desire cut costs prevailed. Cutting down cost was viewed broadly: areas where costs can be cut must not be spared. The attention focused to Oracle iLearning, which to the manager’s view can be easily replaced by an open source system. After all, in this manager’s point of view, most of the support systems that they use within their department were open source. Why not an LMS as well? (Appendix I: Case D, Item 15).

As discussed in the previous section, there were plenty of breakdowns in system use, especially in the later part of the implementation. On top of these breakdowns are the deteriorating system performance exemplified by lagged access and time outs and lack of user-friendliness (Appendix I: Case D, Items 16 through 19 and 21 through 23) As resolution to these issues, the prevailing mindset in the team reinforced by the manager’s priorities, especially among those who were based Amsterdam, was to replace Oracle iLearning. A new LMS system is needed and perhaps indeed an open source LMS can solve those problem – was the unspoken wish of the Amsterdam based team (Appendix I: Case D, Items 21 through 23).

Aside from replacing Oracle with another LMS, there is also the growing tendency to look for other software solutions outside Oracle. This is reflected in impact relation: POST_BP_7 → POST_SP_5 (Appendix I: Case D, Item 20). The difficulty in configuring an assessment, making it up and running in Oracle iLearning prompted the training developer to consider other e-learning authoring tools. While these tools also support the creation of assessments, it is not the ideal way of developing assessments because it takes a longer process to publish them in the LMS.

Last but not least, in fact the most crucial M7 impact relationship is the instance in which the problem of economic survival prevailed over software implementation (Appendix I: Case D, Item 24). In this impact relation, the problem of not generating any software license sales and therefore little of no of demand for training at all makes a learning service infrastructure such as Oracle iLearning redundant. Further, with the dissolution of the training department, Oracle iLearning is no longer needed. Its subscription need not be renewed.

8.7 Discussion

This chapter presents the fourth and final case study in our research, which makes our case study design complete. It is an interesting case of a groupware implementation lifecycle ending in abandonment. Ultimately, it is a failed implementation. The system was decreasing in popularity to its users due to deteriorating performance and poor usability. At the same time, the organization and the team as a whole had shifted to requirements that no longer accommodate the current LMS. We would like to conclude this case study with the following: feedback to the requirements evolution framework (Section 8.7.1) and lessons (Section 8.7.2)

8.7.1 Feedback to the requirements evolution framework

There are several things we have achieved in the completion of this case. First of all, methodologically speaking we have fulfilled the requirements of our case study design (Section 3.3). Secondly, the case is additional confirmation of the usefulness of our requirements evolution
framework which is supported by sufficient examples and materials.

The dynamic of requirements evolution is characterized by seven types of impact relations that we have been able to identify so far. Various instances of these most common impact relations have been found in this case. It is interesting to note that several impact relations represent the case and its implementation in a quite signified manner. For example, in the later phases when the decision to terminate the system can is represented by the impact relations M1: Business problems resolved by business solution and M2: Business solution motivating a software product idea. These impact relations embody the financial crisis as a problem besetting the economic viability of the organization which is resolved by a series of organizational structuring to cut down costs. Among these include dissolving the training department, which ultimately led to the abandonment of the LMS. The resulting product concept is a software product concept that negates or invalidates the necessity of the system in the organization in the light of the circumstances.

Aside from the previous example, the case also provided vivid examples of M4: Software solution specification supported by business solutions impact relations that show efforts taken by the implementing to overcome the shortcomings of the system to meet their needs. While some of the actions taken are rather cumbersome, for example the use of MS Excel to process course evaluation results, which was not really needed. The goals of having evaluation results and have them communicated to stakeholders are much more important and have the priority of being realized.

The organization’s utilization of external consultants provides a good example of M5: Business solutions leading to improved software solution specifications impact relations. Such decisions evidently led to productive results in a way that the training team was able to articulate their software requirements even though it did not have the capacity and the expertise to do so.

Interestingly, this case showed also a dynamic side to within-domain evolution, in which a requirement leads to an update by way of another requirement in the same domain. It is a similar impact relation mechanism but this time taking place in the same domain. This mechanism is observed particularly with the business solution and the software solution specification domains. For the business solution domain, we have observed that the formulation of one business solution statement is supported by another business solution effort. This interaction brings about evolution in the business solution domain communicating a reinforcement mechanism among requirements. For the software specification domain, this case differentiates itself from the other cases by the availability of written requirements, not only for the implemented system but also for the new system meant to replace it. A comparison of these two written specification for the same software class indicated no significant change in requirements. The demand for functional requirements remains the same. The difference lies in the level of elaboration, in which requirements articulated in a later period contain more details about the same functionality (i.e. assessment functionality see Section 8.5.4)

8.7.2 Lessons learned

This case provides a good example of how business requirements drive IT priorities. The eventual choice for and implementation of Oracle iLearning were driven by the growth opportunities resulting from the positive acceptance of the FDS software in the market. The same, although in the opposite context, can be said about the decision to replace the LMS with an open source system and the eventual abandonment of the system. It appears that factors tied to the economic well-being of the organization play a crucial role in system implementation. Therefore, it is important in the analysis process to articulate and pay attention to requirements that can contribute to the economic viability of the organization. Every system that is implemented, it should be tied to some economic benefit for the organization.
Implied in our discussion of M2, M4 and M5 impact relations are the presence and role of people. By this we mean key stakeholders in the process who had the affinity and aptitude for ICT applications. Basically, it was the training manager who came up with the idea of using MS Excel for processing course evaluation results. The same manager was also responsible for the engagement of third-party consultants which resulted into a training curriculum, an RFP, and an LMS system. Creativity and a predisposed orientation towards solution helped in sustaining the implementation. Apparently these properties can be said of the organizational culture within FDS as well, given its line of business as a software development company offering solutions to complex problems in financial data management.

The case also showed how organizational change, i.e. internal restructuring, is closely intertwined with requirements change. It impacts for example how priorities are set and ordered. Appending the training team to a larger team with a different set of priorities had relegated the training operations to a lower priority departmental function. Resources allotted for training were, as a result, minimized. Requirements change, as a result of organizational change, also has its social and human dimensions. For example, the abruptness of the change had alienated some members of the team who stayed in the home office and the rest worked in the Amsterdam office. A most obvious side effect of the reorganization was the power struggle between the new manager and the original training team. The original training team doubted the capacity of the new manager as a training manager given that he didn’t have any experience with training. This resulted into a lack of support for the new manager’s initiatives. This way, the failure of the open source LMS project to push through can be attributed to this. Organizational changes are tricky situations for which no one solution fits all. However, in this case, it could have helped if the transition was given more time and that power struggle should be minimized. Managing requirements change therefore can benefit from organizational interventions that focus on addressing conflicts. Conflicts among team members pose as risk to project success. In this case, the power struggle led to conflicts and lack of support to manager’s initiatives. Power struggle can be minimized by distributing power and autonomy to the original team. In addition, the new manager can try to acknowledge the operational autonomy of their original training team instead of imposing his own wishes. To begin with, the new manager could have recognized that the original training team was a functional and productive team before the restructuring. There was a prevailing feeling that there is no hurry to change that.
CHAPTER

9

MULTIPLE CASE ANALYSIS

Requirements evolution patterns

In this chapter we combine all the requirements and requirements evolution statements derived from all cases with the goal of identifying common mechanisms that represent various patterns of change. Those mechanisms are usual practices in system implementation which ultimately reveal requirements evolution patterns.

The cross-case analysis begins with a recap of all the cases and a brief review of the methodological choice that led to this selection of cases. This is followed by a short discussion of the analysis roadmap based on the conceptual framework which provides the structure for the rest of this chapter.

9.1 Case studies overview

We conducted a total of four case studies (Chapters 4-5; 7-8) based on a case design methodology that calls for 2 similar cases and 2 contrasting cases, with each pair representing worst-case and best case scenarios of system implementation. Figure 9-1 is an overview of the cases and their function in the case design.

![Figure 9-1. Case studies overview (same as Figure 1-2)](image)

Case A and D are classified as worst scenario cases because they represent system implementation with that are both discontinued or abandoned, therefore a failed implementation. Both are grassroots initiated implementation. While far from being considered successful
groupware implementations, Case B and C on the other hand show examples of groupware applications that are embedded in core processes, which by doing so become inevitable. They also remain in continued use. These features contrast the last two cases with the former case study pair (Case A and D).

9.2 Multiple case analysis protocol

We will make use of the same analysis protocol applied to each case study in this multiple case analysis. The only difference is that supporting data will come from all the cases. The updated conceptual framework in Chapter 6 and renewed list of questions (6.4.2) are used to structure this multiple case analysis into the following components: requirements domains (9.3), within-domain evolution (9.4), impact relations (9.5) and breakdowns and initiatives (9.6). Specifically, the pattern seeking questions for multiple cases (6.4.2; L3Q1 to L3Q3) are addressed through a closer scrutiny of data (9.6) and a discussion (9.7).

9.3 Requirements domains

The discussion on requirements domains is composed of six parts: an overview, an individual discussion of the 4 requirements domains and a pattern discovery analysis.

9.3.1 Overview and observations

As the compiled data from the cases show, the requirements domains are all represented with varying manifestations over time (Appendix B). Taking the domains as a whole, we have the following overall observations:

- All requirements domains are persistently represented in all the cases. They are represented in various phases of implementation but not all in one phase at the same time. This implies that the concept of requirements domains is a broad enough framework to capture changes arising from technology implementations that impact requirements. On the other hand, not being represented all at the same time one phase reflect the different priorities and dynamics that take place in a certain groupware implementation.

- The requirements, in total, show an increasing trend over time indicating a continuing increase in requirements as the implementation unfolds. Such trend can be attributed to the fact that most implementations, as reported in each case study, begin with few or no requirements. Usually, these few requirements are top-level specifications and are very general. They needed further elaboration and re-specification as the process further evolves. At the same time, new requirements arise as the software gets used.

- The software solution specification domain is the most represented requirement domain (Appendix B.4). This is of course expected given that software solution specifications as requirements are explicit details and precise enough articulation of business problem specification or a business solution specification. As implied in (2.7.1; 3.3.1, Table 3-1), a software product concept can be executed by one or many software solution specifications.

- Next to software solution specifications, the business solution domain is the next most represented requirements domain (Appendix B.2). This domain represents organizational process changes, tasks restructuring and operational measures taken to not only to adapt but as well as to adopt the software. This observation suggests that system implementation is not only about software but also about the efforts needed to prepare the users and the system to the new situation.

- Between the software and business requirements domains, the software domain, i.e. software
problem and solution together, is slightly better represented, if not evenly, in all cases than the business domain. The prominence of software over business in this particular finding is supported by the earlier observation on predominance of software solution specification statements (3rd bullet statement).

- Between optimal and non-optimal cases of groupware implementation, it appears that the situation in a non-successful implementation leads to more business problems next to software solution specifications. Whereas successful implementations represented by Cases B and C on the other hand yield more business solutions next to software solution specifications.

- The software product concept progressively evolves over time in all cases with the exception of Case A. It seems that in implementations in which the system is more or less intensively used and for a longer period, the users and the context are in an active state of (re)discovering purposes for the software. Prolonged interaction with the software provides the opportunity to get to know the software and therefore think of ways on how to use it, expand user base and find possibilities to solve other problems.

9.3.2 Business problem domain

The domain of business problems proved to be a useful category to capture, generate and discover requirements about business goals, user goals and desires, and most importantly, the higher management context that regulates the groupware implementation. From an overall inspection of the compiled business problem requirements from the cases, the main conclusion that we can draw is that the business problem domain is valid requirements construct. Requirements instances representing business problems can be found in all cases with various representations over time. A list of all business problem statement gathered from cases is shown in Appendix B.1. A few things can be said about this compilation and its characteristics:

- Except for Case C at pre-implementation, business problem domain requirements can be found throughout the implementation process. (Appendix A)

- There appears to be no clear pattern of progression of business problem requirements over time (Appendix A). What we can say is that for Case A and B, business problems are predominant at the beginning of implementation (pre-implementation). For Case B and D, they peak at the latter part of implementation (post deployment). It is interesting to note that the business problems at post implementation are composed of breakdowns in system interaction. As users get to use the system more intensively and purposefully, the interface problems become more apparent.

9.3.3 Business solution domain

Through the business solution domain, we were able to capture organizational as well as individual measures taken to address business problems and likewise support groupware implementation. On the basis of the list of business solution statements given in Appendix B.2 we add the following observations about this domain:

- In a pattern similar to that of the business problem domain, the business solution domain is also represented in all the cases but in varying representations over time. For example, in our compilation, the business solution requirements for Case B started to pick up a little later in comparison with the other cases.

- Over the course of implementation, our compilation also shows that the business solution domain continues to thrive. This indicates that the implementing organizations continue to adapt to the software regardless whether the implementation is successful or not. People are
more flexible than software is.

9.3.4 Software product concept domain

Also, the software product concept domain which captures the mental models associated with the groupware application is a requirements domain that thrives in the course of the implementation. This thriving can be inferred in the requirements list in Appendix B.3 and we add the following observations:

- The software product concept domain is a ubiquitous requirements domain in all cases. It is also present in all phases of implementation in optimal groupware implementations (Case B and C). However, in Case A and D, there are periods in which we did not observe a representative software product concept: early implementation in Case A and pre-implementation in Case D.

- There is marked increase in the generation of software product concepts way later in the implementation time frame, peaking at post-implementation. This gives an indication that implementation is an on-going process. It is also interesting to note that in the middle phase of implementation for Case A and B the software product concept domain is at standstill, slightly indicating that stakeholders lie low on their appropriation of the software. This phase can be seen as a transition period in which stakeholders try to internalize what the software can mean for them on the basis of an initial concept and eventually figuring out a more specific purpose for the groupware.

9.3.5 Software solution specification domain

This domain, as expected and already elaborated is the most represented requirements domain with the following features on the basis of the cases:

- It is ubiquitous not only in all cases but also in all phases of the implementation. This means that functional, quality as well as system specifications are already known as early as possible. This reinforces the claim that software engineering does not necessarily follow a waterfall model in a way that specifications are already defined even when there are no problem specifications yet.

- On the basis of individual cases, there is no apparent growth in specifications over time that can be derived (Appendix B.4). However, on the basis of the consolidated specifications gathered over time, we can say that software solution specifications increase from early implementation to post implementation.

9.4 Requirements evolution: within-domain evolution

The concept of within-domain evolution extends our understanding of requirements evolution as change in requirements domain over time. As already known, we speak of four types of within-domain evolution whose individual evolutionary features will be discussed individually (see Chapter 6).

On the basis of our earlier analysis of each domain and a summative examination of the requirements listed in Appendix B, the following can be said about within-domain evolution in general:

- All requirements domain evolve over time – be it in the form of additional or shifting requirements from one period to the other. However, the progression of change per domain and per period varies (Appendix B).
• The changes in a requirements domain from one period to another is not always incremental; they vary from period to period and from domain to domain. While we say that there is a marked increase in requirements over time, this is however an overall impression from all the cases. The progression of each domain from one period to the other is not necessarily of a gradual increase.

• There is no comparable progression in requirements over time for which one can say that there is an established pattern of requirements succession. The progression of requirements per domain differs for each case. The only exception is the software product concept domain in which most cases have shown a continued increase in the number of requirements over time. Of course, evolution is broader than a change in quantity of requirements that can be gathered over time; the volume of requirements can remain constant over time but the meaning or essence of the item change over time. That is also a form of evolution. However, in this situation, what we would like to point out is that in both cases, the increase in the volume of software product concepts indicate that new ideas or meanings are continuously being generated about the software over time.

• Among the different requirements domains, it is the software product concept that progressively evolves over time (Appendix B.3). This is the only requirements domain that has been observed to continuously change incrementally throughout the implementation. This observation is in line with Ciborra’s concept of ‘drift’ (Ciborra 1996) and Orlikowski and Hofman’s (1997) ‘emergent and opportunistic’ models of organizational change resulting from groupware implementations. Ciborra (1996) described the shifting purposes for which a groupware application is implemented as ‘drift’, giving the example of a Lotus Notes implementation in which the purpose of use shifted from a communication tool to a knowledge management system, and to a speculated future use as a project management system. Drift is reflected in the changing mental models of the software through evolving software product ideas. Likewise, the continuously evolving software product concept observed in the cases capture both emergent and opportunistic changes. An example of emergent change is illustrated in Case B in which the mental model of a closed system tacitly developed as teachers made use of FCC to communicate electronically with colleagues but not with students. Such perception during early use is prolonged to further shape an image of the software as a virtual workplace and to foster the idea of a closed system -- only those in the address book are part of the system (see Appendix B.3: Case C requirements statements). This view is reinforced by the system’s simplified address book functionality. Case D provides a good example of opportunistic change when the learning management system (LMS) was used to mitigate a potential rift with customers and the internal sales and account management team with respect to the provision of training services. As originally intended, the LMS was meant to support the delivery of training as it takes place in the classroom. Eventually, the LMS application became a useful tool not only for the training team but also for the sales and account management departments who have found a solution for onsite training requests that cannot be immediately satisfied. The system became useful in managing customer expectations. Apparently, for the new training manager, this mental model of the LMS as a sales and account management tool is the most important feature of Oracle iLearning that he associates the system with. On the other hand, software projects in aircraft and automotive industries are not expected to have this behavior. Concepts of drift and models of organizational change are further discussed in the conclusion of this chapter. See Section 9.8.2.

• Specifically, for Cases B and C, which represent successful groupware implementations, the software product concept progressively evolves over time. The groupware application in these contexts continues to assume additional product identities in the course of time. On the other
hand, this is not observed with the other case studies representing non-successful groupware implementations. The software product concept does evolve but not progressively over time.

- It is also a refinement process in which a requirement is updated by another requirement in the same domain (see third bullet point). The business solution (see 9.3.2) and software solution specification (see 9.3.4) domains provide detailed examples of this refinement mechanism.

9.4.1 Business problem domain evolution

Aside from the general observation that the business problem domain evolves over time, this evolution has several interesting characteristics (see also Appendix B.1):

- New business problems that arise represent a shift in the level of content – from general to more specific goal statements. For example in Case A, pre-implementation business problem statements represent higher management goals which cannot be directly linked to the software. In later periods, the goal statements become much more focused, i.e. an intensively used knowledge exchange system. Or in the case of Case C, the business problem statements shift in content level is from generic to more operational task statements for which a software function can match.

- With the shift in content level comes also shift in stakeholder representation and involvement. As stated in the previous finding, at pre-implementation, the business problem statements can be mostly attributed to higher management. Further as the implementation progresses, new business problems are issued by users and these usually pertain to specific tasks that they cannot do with the software or features that they do not fully understand how to use.

9.4.2 Business solution domain evolution

Through the business solution domain and its representative requirements instances (Appendix B.2), requirements evolution can be characterized as:

- An evolution of business solutions without software towards business solution revolving around an implemented software product. This is of course when the business solution merits or calls for a software implementation in order to realize its benefits.

- Increasing stakeholder involvement, with end-users becoming more articulate due to prolonged exposure with the groupware application. There is also a shift from a top-down solution approach towards bottom-up (see Appendix B.2: Items 2 and 22 as examples for both observations).

- For non-successful implementations in comparison with successful ones, it appears that at post implementation phases, suggestions to improve system usage seem involuntary and entail extra, non-motivating work (Appendix B.2: Item 19).

- Over time, as software use intensifies, the business solutions at the level of stakeholder proponents give way into policies and process change further expanding software use (Appendix B.2: Items 7 and 21) and at individual level, users adapt their behavior and way of working order to accommodate the software (Appendix B.2: Item 24). Therefore, the overall impact of changes in the business solution domain are both at the level of the organization and of the individual.

- Refinement mechanism indicating a process of reinforcement and support. For example, in Case C, the mandate of using TeleTOP university wide (Appendix B.2: Item 2) was reinforced by taking specific measures such as installing a TeleTOP expert in each faculty (Appendix B.2: Item 9).
9.4.3 Software product concept domain evolution

As we have already mentioned, the software product concept progressively evolves over time (see Appendix B.3). This form of requirements evolution can be characterized in general as follows:

- Continuously changing software product for the same software regardless of implementation outcome. In other words, throughout its lifecycle, a groupware application can assume multiple product concepts without necessarily changing any of its architecture of packaged functions and specifications. This we have seen in most cases. Case A is about a knowledge management system which, due to its low level of use is suggested to be transformed into an expert system and eventually into a project management system. The groupware application in Case B is initially conceived as an e-learning application which eventually became a collaboration and communication tool among teachers. Case C is likewise initially thought of as an e-learning to support the teaching and learning process. However, its de-facto use came to be as a course management system, which teachers intensively make use of. Case D is a similar example of an e-learning concept conceived of a system which evolved into a learning management system supporting blended-learning and eventually into a sales and account management support tool. In all these groupware tools, the mental models about the system continue to change but the software and its structure remain the same. Of course, in the case of Case A, the transformation towards an expert system implies an expansion in functionality, and that of Case B, which is a COTS product which gets upgraded from time to time, the core architecture of the tools remain the same.

- The shift in product software concepts cover a broad range of product labels: from knowledge management system to project management and expert system (Case A); from e-learning to an administrative communication and collaboration tool (Case B); from a tele-learning tool to a classroom support system (Case C); and from a ICT support for training into LMS support for blended learning and finally as a sales and account management tool (Case D). This observation further reinforces the known characteristics and outcomes of uncertainties in implementing general purpose software such as a groupware. The use and outcome of implementation cannot be anticipated in advance (Ciborra 1996, Orlikowski 1996; Orlikowski & Hofman 1997; Pipek & Wulf 1999).

- The software product concept evolution is not only about assigning a product identity or label to the software as a whole. It also speaks about an evolving or developing mental model about a particular software property, what this enables and disables, and making generalizations about then software in general, i.e. an exclusive, closed-system software (Appendix B.3: Items 8 and 9).

9.4.4 Software solution specification domain evolution

The software solution specification domain evolution is about the changes in the specifications of the implemented software over time. The conclusions we can make about this domain (Appendix B.4) on the based on the cases are:

- Software solution specifications continue to arise throughout the implementation process.

- It is possible to know specific functions the groupware must have before the implementation takes place. For some cases, a substantial number of specifications are already available at pre-implementation, an observation that is consistent with the waterfall model of software development. The same specifically applies when the implementing department is simply following through an earlier implementation of the same software.
• There is an apparent increase in the participation of more stakeholders, specifically of end-users who voice out frustrations about the software and suggest ways of improving ease of use (Appendix B.4: Items 47 through 49).

• On the basis of the previous conclusion, the evolution of software solution specifications over time also represents a shift in focus. Specifications during pre-implementation are purely focused on the software but this shift towards a more inclusive set of specifications that take into consideration user reaction and feedback (Appendix B.4: Item 49).

• Software solution specifications in Case A and Case D appear to continuously grow over time across periods. While Case B and C, cases of optimal implementations, indicate a slight waning off of specifications in the latter phases. It is interesting to note that in comparison with the other cases, the statements during post implementation of Case A appear to be artificially prescribed, superfluous and vague (Appendix B.4: Items 15 through 24).

• Just like the business solution domain, this domain also demonstrates a process of refinement in the form of improvement and functional correction. This change indicates also software evolution. For example, Case C, the workspace function of TeleTOP was reportedly not working properly, i.e. it did not support large files upload and it does not sufficient storage space. Accordingly, this error was corrected with a new version of the workspace, which was made available for use (Appendix B.4: Items 41 through 42).

9.4.5 Refinement process

Our analysis of within-domain evolution lead to us to a common mechanism of change noted in the discussion of business solution domain evolution (9.4.2) and software solution specification domain evolution (9.4.4). In both discussions, we have referred to this domain evolution mechanism as refinement. It is a process in which the domain expands in resolution to a requirement belonging to the same domain. In a comparable principle to that of an impact relation, refinement is the resolution of a requirement in the same requirements domain. Therefore, requirements evolution in the form of within-domain evolution can take the special form of a refinement process.

9.5 Requirements evolution: impact relations across domains

Impact relations are requirements evolution mechanisms brought about by the resolution of breakdowns and initiative that usually take place across requirements domains. There are 7 types of impact relations that have been identified (Table 6-1). We will go through these impact relations one by one and discuss each using pertinent data provided by the cases. A compilation of the impact relation instances are given in Appendix C to I. The goal is to identify patterns of evolution revealed in each of these impact relations or a combination thereof. Next to this, we will also rename each impact relation to make them recognizable. Likewise, the patterns that we will discover as common occurring mechanisms will be identified with familiar names or phenomenon.

9.5.1 Software triggered business problems resolved by business solutions (M1)

The impact relation M1 can be aptly renamed as software triggered business problems resolved by business solutions especially in the context of a software implementation project. The impact relation is observed in all of the cases in this study and in all of the implementation phases (Appendix C). We can identify two mechanisms in place: (i) business problem expansion and (ii)
software problem expansion.

Business problem expansion

In the first iteration, M1 captures the management action that takes a top-down approach in a software implementation project. First, problems are defined, expressed and communicated. Second, management and organization-oriented approaches are sought at the strategic and operational level. Case A demonstrates this approach as the top management level thinks that a problem exists in the integration and mergers. Strategic, non-software solutions are sought in the form of a knowledge management strategy and re-organization (Appendix C: Item 1). The same goes for Case B, which is a successful implementation. There is also a top-level rationale, embodied in the desire to innovate educational process. It provides the motivation to go for the operational strategy of replacing an email system, in order to fit the top-level strategy. Various examples of this mechanism can also be found in Case D in which the creation of a training department is a response to address growth and business opportunities.

Software problem expansion

The next iteration is when the implementation proceeds and new sets of problems arise because of the existence of the software. This can be observed in the link between the desire to improve the usage of the system and the suggested means on how this can be done in Case A (Appendix C, Item 3). The system was a source of problems for the proponents because it was not being used as expected. Because of this, the proponents were not able to achieve their goals. In order to improve the situation, it was suggested that target users, i.e. product managers and actuaries, should upload training materials that they gather from seminars and trainings to the system. This should populate the contents in the system and therefore make it more attractive to use. For Case B, the long-term use of FCC further exposes the users to what they don’t know about the system by using it on daily basis with difficulty. This problem encountered by the users is sought to be addressed by organizing training sessions about the application for the users (Appendix C: Item 13).

The multiple occurrences of this impact relation over time in various stages of the implementation offer alternatives explanations of system implementation other than a top-down approach to system implementation (see 9.6.3, second paragraph). The top-down approach entails that implementation should begin with the problem statement and then resolve it first with a business oriented non-software solution. This approach presupposes that an impact relationship such as M1 should only occur in pre-implementation. While our observations confirm that such a top-down impact occurs in the early stage, there are other instances of M1 as well, which take place long after the software has been implemented. This phenomenon represents the evolution in an implementation, in which the software brings about environmental changes that could lead to a new iteration of problems with a specific focus: the software and its use. This is supported by our observation that most of the M1 impact relationships late in the implementation are breakdown-driven.

9.5.2 Software product (re)definition (M2)

That the focus of our research is all about the implementation of a software tool is reflected in our findings about the impact relation M2 which is represented in all cases (Appendix D). M2 captures the transformation of business solution strategies and business policies into a software idea. This impact relation also shows how social and behavioral adaptations can affect the mental model of what software-in-use should be. This is instantiated in Case B in which the policy of replacing and upgrading the email system
presupposes software that offers an enriched set of functionalities for enabling communication and collaboration (Appendix D: Item 4). Users in Case B were used to a simple email system with no added functionalities such as calendar and shared workspaces. In Case A, the requirement which states the adoption of a knowledge management policy as a strategy became amenable to a software product, with a corresponding concept of a knowledge management and network system (Appendix D: Item 3). Such conversion of an action plan into a software idea is an attempt at software product definition. Software product definition is identifying a software product concept that would extend and augment a business solution specification such as policies, re-organization efforts, and procedural changes is a form of product development pattern. It is similar to brainstorming and market research efforts that seek to generate ideas for a new product. Inputs from the potential user environment provide ideas for a new product concept.

**Redefinition**

Just like M1, M2 is observed to take place multiple times at various phases of implementation. Specifically, M2 is observed to be most visible during early implementation and eventually in the post-deployment period. The early occurrence of this impact relation is consistent with the idea of a top-down approach to implementation in which a business solution is supposed to precede a software choice. However, M2 features also another iteration cycle as the implementation proceeds inwards focusing on software in the post deployment period of implementation. In such iterations, the resulting business solutions serve to re-define the implemented software’s product concept. For example, we can see in Case B that in the long-term when the software gets more intensively used, ideas are brought up on how the existing software can be used to improve various processes. This expands the mental model of what the software is about and what can it be used for (Appendix D: Items 8 through 10). The same can be said in Case D in which the idea of having a learning management system (LMS) in the training department will expand to become a tool that addresses the problems of sales and account management department (Appendix D: Item 19). Relating this to the earlier of notion software product definition, at the later stages of implementation, this impact relationship also captures the essence of redefining the same software product for a different purpose.

It is interesting to note that most of the requirements in this impact relation are initiative-driven. Overlaying the findings of each case also highlight the differences between successful and non-successful cases. For non-successful implementations such as case A, we have observed that M2 does not go beyond early implementation. This can be explained by the fact that for a non-successful implementation especially in the later phases, the focus of activities and efforts is more on the problem space than the solution space. In Case A, there is an additional explanation for the fact that we didn’t find M2 impact relations in post-deployment. In this phase, the software is about to be retired and replaced. The implementation is then terminated and in case of a software replacement, a new implementation is in place.

**9.5.3 Classical specification (M3)**

M3 represents the link between product concept and its realization through a certain set of software specifications. The specifications make the software product concrete and tangible to users. This is classical specification in software product development. For example, the concept of a knowledge management and exchange system in Case A is realized among others by the specification that says it should store data items and that it should provide a means of categorizing data (Appendix E: Items 1 and 3). In Case B, the concept of a communication and collaboration tool is realized by specifications such as having the facility for promoting online communication through email and a shared workspace.
environment (Append E: Items 18 and 19). For Case C, the product concept of a tele-learning tool such as TeleTOP is realized through its specifications as a finished product with core domain functionalities such as course roster, presentations, shared workspace, etc. (Appendix E: Item 20). For Case D, the simple concept of having an open source system to replace the existing LMS made the specifications for a new system a lot easier (Appendix E, Item 38).

The occurrences of impact relation M3 also show that this type of requirements evolution is present in all phases of implementation. However, it is observed to be most relevant during pre-implementation or early implementation, which is quite expected in a software implementation project. There must be something tangible and usable before a software implementation officially gets going. On the other hand, the occurrences of M3 show that a software product is not completely finished upon implementation. The continued interaction with the software brings about ideas on how to enrich the product concept by improving its specifications. For example in Case A, the shift in the software product concept from a knowledge management system towards an expert system requires the connectivity with existing expert systems in the non-life insurance business (Appendix E: Item 14).

9.5.4 Workarounds (M4)

M4 is an example of a flow that goes against the expected sequence of events in a top-down approach to implementation (see Section 9.7.3). M4 occurs in a situation in which a software application upon its deployment does not immediately bring about positive results and needs additional stimulation. Therefore, workarounds are needed. These workarounds were observed in all of the cases and at various phases of implementation. In Case A, there was a complete breakdown; it was apparent that the initial implementation did not lead to the expected results and therefore the software needs to be improved. This led to the move that involved the users in redesigning the software by coming up with their own classification rules (Appendix F: Item 1). M4 is also a means of representing ways in which users cope with the system. This is illustrated in Case B in which quality attributes relating to usability and interface design bring about complaints from users which in turn are resolved by users on a case to case basis either by taking action or none at all (Appendix F: Items 5 and 6). In Case C, when the teachers began to complain about the tedious process of filling in the roster in TeleTOP, the faculty provided support in the form of student assistants who supported teachers in setting up courses in TeleTOP (Appendix F: Item 10). In case D, the limit imposed by the subscription on users and server space prompted the training team to seek creative ways on how to provide onsite, i.e. by providing USB sticks (Appendix F, Item 17) and by changing the way how work is done (Appendix F, Item 16). Apparently, Case D provides the most number of instances of workarounds among the cases.

M4 took place in two implementation periods, namely early and post-deployment. Not surprisingly, it is absent in pre-implementation; it is an adaptive measure, meant to improve software utilization, which will only occur after a functioning software system has been implemented. M4 was always observed to be breakdown-driven: problems with the use of the software are addressed by adapting the business procedures.

As a familiar pattern of action, workarounds are efforts to overcome the imperfections of a software solution by making changes in the user environment to accommodate the software. For example, the system is confusing; therefore we define procedures so that the task can be carried without much hassle. Explicit instructions have to be made on how to handle certain software features so that people don’t get confused (Case C). All told, these are workaround aimed at getting over breakdowns.
9.5.5 Business solution elaboration (M5)

The impact relation M5 captures the many ways in which a detailed business specification can provide useful inputs for what the software should actually do in terms of specific functional and quality attributes. The way in which business solution specifications improve the formulation of software functions and quality attributes as indicated by the impact relation M5 can be summed up as elaboration. By enacting informative policies and executing actions with the software in mind, specifications get more detailed, and the software is brought closer to the users. The compilation of all M5 impact relations from the cases highlights two significant findings on how business solutions can improve the formulation of software specification. Firstly, when specified early and properly focused on the software, business solution specifications can lead to generation software solution specification. This is the desirable outcome because the software is brought closer to the users. For example in Case A, the design exercise involving the end-users led to the generation of meaningful specifications (Appendix G: Item 13). Secondly, working around the limitations of the tool, i.e. the case of the limited space in TeleTOP point our errors in design which could be corrected (Appendix G: Item 16). Thirdly, involving external support, i.e. consultants and interns to focus on the software domain in question can also help in formulating the necessary specifications. Case D is a good example of this in which the involvement of third party consultants resulted into led to the formulation of an RFP (Appendix G: Item 18) and the use of an intern work on the open source project facilitates the production of specifications (Appendix G: Item 27).

Altogether, the various elaborations triggered mostly during the early phases of implementation capture the efforts taken to support the implementation as it unfolds. For example in Case B in the period immediately after the implementation, the policy of having all new employees learn FCC in 14 days indicates that the software should have as a quality requirement the usability attribute of being learnable within two weeks (Appendix G: Item 15). This is also why, based on our observations that elaborations are usually initiative driven. The latter is consistent with the fact that while an M5 relation indicates an interactive relationship between users and software, the stream of developments proceeds from users towards software.

9.5.6 The software solution becomes the problem (M6)

This impact relation encapsulates the situation in which the implemented software product (read: final solution) actually brings about new problems. In other words, the software solution becomes the business problem. This is best illustrated by Case A, in which the outcome of the system’s implementation has turned into a disaster. On hindsight, this impact relationship is one way to summarize the inadvertent results of Case A. Even when new software solution specifications are being sought at post deployment, all these are but indicators of dissatisfaction with the tool (Appendix H: Item 1). It is for this reason in this case that the software solutions in the post-deployment period are linked to the business problem domain. This link is strengthened by the fact that most of the software solutions specifications at post deployment are no longer about improving the current system but are rather hinting at a new or a different system. These, in turn point to a new business problem.

M6 is also about problems relating to breakdowns in system use due to poor system usability and decline in performance. This is epitomized in Case D in which the system – when it does not function as expected, it becomes a hinder in an otherwise smooth flowing process (Appendix H: Items 20 through 24). The system becomes a source of breakdowns. This kind of relationship is also present in successful groupware implementation such as Case B. In Case B, this impact relation depicts the specific side-effects of implementing of a software tool. The implementation
of FCC shows that while generation of a productive software that has the function and quality to support work, it also opens a whole new set of problems at the user task execution level (Appendix H: Items 12 through 14). For example, the fact that software such as FCC is available 24/7 also results in teachers working continuously outside office hours and on weekends. This pattern is a constant reminder that software is imperfect; it does not offer a quick and complete solution and breakdowns are expected to happen along the way.

9.5.7 Software induced business goals (M7)

From top-down perspective, M7 seems to skip one step: from a business problem directly to a software product concept, without an intermediate business solution. However, our multiple case analyses conclude that this happens occasionally in software implementation projects. Business goals are induced by software.

A consolidated inspection of all the impact relations of this kind says two things about what happens in an implementation effort and the way decisions are taken. Firstly, the existence of an M7 phenomenon highlights a prominent practice in system acquisition and implementation wherein problems are worked out and resolved by iterating on software product concepts. Seeking solutions to problems resulting from the regular functioning of an organization appears to be closely intertwined with a software choice. A business problem is defined in terms of a matching software in mind; it is wishful thinking. For example, in Case B, the problem of introducing an ICT innovation that is aligned with the general domain of operations of a school namely, education leads to an e-learning software product concept (Appendix I: Item 4). Secondly, a business problem can be matched with one more or software product concepts which is true especially in the later phases of implementation. In later phases of implementation when new business problems arise due to software implementation, new software product concepts arise for the same software. In other words, the same software can assume multiple product concepts as the business problem domain keeps on changing. For example in case A, the problem of not generating sufficient usage for the software is aimed to be resolved by re-packaging the knowledge management tool into project management system (Appendix I: Item 1). An important feature of this impact relation between the business problem and the software product concept domain is that while the software product continues to change, driven by business concerns, the implemented software in question does not necessarily need to change. In another example, Case D, the instances of M7 impact relations – all in the post-deployment period, are all about theme of ‘we are not happy with the current software, we need to replace it with another one’ (Appendix I: Items 15 through 23).

In all cases which this impact relation is observed, the actual implemented product remained the same. Or the requirements remained the same. This is true of Case D even though most of the M7 impact relations point to a new software product concept. When we look at the specifications from the RFP to the open source LMS project, the real requirements remain the same (see discussion in Chapter 8: Case D, Section 8.5.3.). Both specifications refer to the same software product concept, with a slight difference in implementation. One is a hosted application and the proposed new application is a open-source. However both are learning management systems (LMS) in product packaging.

Therefore, our conclusion remains that evolving product concepts triggered by business problems need not necessarily result in new software and that a software product can have multiple products concepts.

9.6 Impact relation patterns

The consolidated analysis of impact relations from the cases bring to light several impact
relations compositions representing additional evolution patterns. These compositions are combinations of impact relations that more or less represent a continuously discrete cycle. Of course, we can say that an impact relation is already a cycle of evolution in itself. However, we do know that in some instances that the process of evolution does not stop at the next domain, i.e. the updated domain in an impact relation. The process usually continues. This is also because the issue, which the requirements statements articulate persists. Earlier, we referred to an issue as an underlying concern which when articulated become requirements statements (4.6; 5.6; 5.7.3). An issue can be a breakdown or an initiative. The impact relation compositions show that an issue can be resolved in various ways. It can be the case that a breakdown is resolved by an initiative which results in yet another breakdown. Or an that initiative leads to additional opportunities. These are elaborated by the 5 alignment patterns that we have come up with:

- Top-down alignment: \[ M_1 \rightarrow M_2 \rightarrow M_3 \]
- Solution-centered alignment: \[ M_7 \rightarrow M_3 \rightarrow M_4 \]
- Opportunistic product recycling: \[ M_4 \rightarrow M_2 \]
- Progressive solution enlargement: \[ M_1 \rightarrow M_5 \]
- Re-orientation: \[ M_6 \rightarrow M_7 \]

9.6.1 Top down alignment: \[ M_1 \rightarrow M_2 \rightarrow M_3 \]

M1 followed by M2 and followed by M3 (Figure 9-2) captures the classical linear progression in a top-down implementation. Not surprisingly, this alignment is observed in the pre-implementation to early implementation phases and as the very first form of iteration. This means that most cases start with a top-down approach and eventually show different alignments or impact relations later in the implementation.

![Figure 9-2. Top-down alignment: M1→M2→M3 impact relation composition](image)

Case A provides a good example of this alignment wherein the following requirements instances are all linked together sequentially per domain; therefore giving rise to the alignment of impact relations \[ M_1 \rightarrow M_2 \rightarrow M_3 \]:

- Business problem: \( A\_PRE\_BP\_2 \). “One of the goals of the organization is to build a community of employees and to evoke a feeling of ‘one AIG company.’”

- Business solution: \( A\_PRE\_BS\_1 \). “The adoption of a Knowledge Management (KM) strategy is a means of community-building.”

- Software product concept: \( A\_PRE\_SP\_1 \). “The system should be a knowledge management system for enabling knowledge exchange.”

- Software solution specification: \( A\_PRE\_SS\_F\_1 \). “The system should store and remember data.”; \( A\_PRE\_SS\_O\_2 \). “The system should be developed in Lotus Notes.”
As the implementation progresses and the problems and solutions get more detailed along the way, this gives rise to new instances of M1 → M2 → M3 compositions. This pattern is either motivated by a breakdown or an initiative.

9.6.2 Solution-centered alignment: M7 → M3 → M4

The sequence of impact relations M7 → M3 → M4 is a solution-centered alignment because the business solution domain is skipped, which in turn becomes the last one to get updated (Figure 9-3). It represents the premature jump from business problem towards software product concept, which can be motivated by either a breakdown or an initiative, but with vague ideas about what the software should do. The flow of this pattern is (i) problem definition or articulation by stakeholders (eventually system owners and users); (ii) resolving the problem by means of a software product concept; this means that the stakeholders already have thought of software as a solution to the problem and with some rough ideas of what the product domain will be; (iii) working out the specifications of the software product into what specific tasks it should perform as well as and functions and data it should have; (iv) finally, the developing policies, re-designing procedures and adapting one’s work behavior to suit the software to the circumstances.

![Figure 9-3. Solution-centered alignment: M7→M3→M4 impact relation composition](image)

This alignment takes place in Case B. The initiative to take part in efforts to introduce educational innovation was met with the idea of implementing e-learning software (M7, Appendix I: Item 4). E-learning was then easily associated as educational innovation. As an e-learning software, it should be possible to support online communication between teachers and students (M3, Appendix E: Item 17). After FCC was implemented with such functionality and after it had proven to support communication among colleagues, a new policy was formulated: teachers should use the groupware for collaboration communication and document exchange(M4, Appendix F: Items 2 and 3).

Basically, this pattern captures the impulse: “We have a problem and this software can solve it for us”. Realizing later on that the software only addresses part of the problem or does not address it properly, efforts are initiated to ‘justify’ the software. Users are prompted to think functionally, to look into tasks and processes more carefully such that the software can be used to further support these.

9.6.3 Opportunistic product recycling: M4 → M2

This impact relation composition spans two relations that closely follow each other: M4 → M2 (Figure 9-4). It represents a situation in which the implemented software leads to changes in business process, tasks, outlook and strategy (mainly to accommodate the software) such that these eventually brings about a new or a redefinition of a product concept for the same software. We refer to this composition as opportunistic product recycling. As we are already speaking about an already being used software product with known specifications, this alignment can be said to
belong not only to the latter phases of implementation but also in a later iteration cycle.

In this particular alignment, the intrinsic property of software is mediated by an accommodating user environment which ultimately leads to a changed software product concept. That is why it is opportunistic: the intrinsic property of the software is transformed into solutions by the environment in which it is used. While the software product concept has changed, the software product did not change at all. This is one of the unique features of this evolution pattern. It leads us to conclude that in a groupware implementation effort, the software product is amenable to changing product concepts, which translate to flexibility in purpose of use. A closer look at cases, in which this alignment pattern is observed, provides support to this finding. For example, in Case A, the expansion of the software capabilities demanded of KennisNet has motivated the involvement of users to come up with their specifications. This led to new product concepts such as a project management tool (M4, Appendix F: Item 1; M2, Appendix F: Item 2). Having gained a better idea of the tool and what it can offer them, the actuaries and product managers have come up with the a project management tool as an alternative use for KennisNet. In the case of First Class implementation, likewise, the intrinsic properties of the software enabled the institution to expand their thinking to make use of the software as a collaboration tool (M4, Appendix F: Item 3). It happened that the software product fits well into the virtual and distributed nature of work in the organization. First Class Client is promoted a medium and a platform for exchanging documents and a direct window to what is happening to the organization. Such policy of use which restructures work creates a mental model that associates the software tool with work and the organization (M2, Appendix D: Item 5). For an individual teacher of the institute working at her own district, the software tool provides a substitute for the organization and her colleagues (M2, Appendix D: Item 7).

As a familiar requirements evolution pattern, opportunistic product recycling is all about reassigning a new software product concept to the same application without architecting a functional change. It is opportunistic because the user environment actively takes part in knowing the software and their needs. Such knowledge stimulates creativity in a way that new purposes for using the same software are thought about. This pattern shows how a breakdown can be turned into an opportunity.

9.6.4 Progressive solution enlargement: M1→M5

This pattern which we observed in multiple instances in Case D is all about taking specific measures which lead to desired outcomes such as detailed software specifications (Figure 9-5) Knowing what a system should have and should do early on in the process help in cutting costs and in hastening the acquisition of software. This pattern implies that prioritizing a business problem and paying attention to it can lead to creative choices of solutions which it makes it possible to skip a product idea and go to directly to specifications. A global product concept is not
necessary in order to arrive at software specifications. Such solution choice for example is
hanging off the solution to another resource instead of pushing one’s own resource. Or in other
words: seeking help from others. The following examples from Case D serve to illustrate this:

- **PRE_BP_3. “The new training department is not experienced with the provision of training
  services and development of training materials.” → EARLY_BS_1. FDS Academy would like
to engage the services of a consulting company for the LMS selection process. A RFP was sent
out with a list of functional requirements. See Table 8-3. Requirements derived from RFP.” →
EARLY [RFP]*. “All statements in the RFP listed in early implementation.” (See M1,
Appendix C: Item 26 & M5, Appendix G: Item 18)

- The use of third party consultants served to address the limitation in the capacity of the training
team. It helped the team in achieving its goal of identifying specific requirements for an
application that could support the provision of training services.

- **POST_BP_5. “New FDS training manager wants to cut down costs.” → POST_BS_6. “The
search for an open source LMS should be assigned to an intern. The intern can also make an
inventory of LMS requirements for FDS Training. See Table 8-5 for the open LMS requirements
inventory.” → POST [OpenSrcLMS]*: “All statements in the open source LMS project RFP
listed in post deployment.” (M1, Appendix C: Item 29 and M5, Appendix G: Item 28)

This example from the case is similar to the previous one, except this time the external help is
acquired through the use of an intern. This action resolves the goal of minimizing costs
compared to getting an external consultant to do the job. The result is the same: a written set of
specification for training support application.

![Figure 9-5: Progressive solution enlargement: M1→M5](image)

### 9.6.5 Re-orientation: M6→M7

The final impact relation composition that we have discovered is a problem-driven software
product concept evolution given by the impact relations M6 and M7 (Figure 9-6). This
composition is initiated by a software solution specification that brings about new business
problems, and these business problems reshape the product concept and mental models about
software attributes. It is all about re-orientating the focus of the software, shifting it to another
purpose because the current purpose cannot be fulfilled or that there are breakdowns in carrying it
out. The following instances from the cases illustrate this evolution:

- Case A: The resulting product specifications (or set of specifications) of the updated version of
KennisNet did not generate the expected level of usage such that the proponents maintain the
hope of still being able to muster user interest for the project (Appendix H: Item 1). Such
problem statement informed by the software solutions specifications give way to a changing
product concepts such as a project management system or an expert system (Appendix I: Items
1 and 2).

- Case B: This composition is illustrated by Case B in an interesting manner. The quality attribute that the FCC has to be learnable in 14 days does not appear to be sufficient to enable users to learn how to operate the system properly. It fosters an inconsistent, if not, an incorrect view of what the software can actually do (M6, Appendix H: Item 13; M7, Appendix I: Item 9).

- Case D: Case D offers a number of breakdown situations in which the system is the source of the problem itself. These include poor system performance and usability (see Appendix I: Items 12 through 7). The gravity of these breakdowns stimulates re-thinking about replacing with the system with a less costly application in the light of the goal to cut down costs (Appendix I: Item 13). The problems encountered when using the system lead to change in the way of thinking about Oracle iLearning and as the M7 product statements show there is a strong bias for system replacement, preferably open-source since it does entail acquisition costs.

![Figure 9-6. Re-orientation: M6→M7](image)

In this pattern, when the intended or implemented solution becomes the problem, a re-orientation period follows. Repackaging the implemented software by repositioning its product concept and reshaping people’s mental models is a way of addressing the situation. Either new product ideas are brought for the same software or a new software product concept is already being suggested. This pattern is an example of a breakdown that has no easy way out.

### 9.7 Mechanics of evolution: breakdowns and initiatives

An impact relation is triggered by breakdowns or initiatives. Requirements evolution results from the resolution of a breakdown or an initiative from one requirements domain to another requirements domains. A breakdown or an initiative articulates an issue which give rise to requirements. A requirements statement is an expression of an issue. To further distinguish these two triggers and their function in requirements evolution, distinctions are made among the impact relations and their corresponding requirements instances. This gives rise to a further classification into typical breakdowns, typical initiatives and part-breakdown and part-initiative situations.

#### 9.7.1 Typical breakdowns

A comparative examination of the impact relations and their occurrences points to M6: Software solution specification leading to new business problems, M4: Software solution specification supported by business solutions as situations largely motivated by breakdowns, M1: Business problem resolved by business solutions and M7: Business problem resolved by (re)defining software product concept (Appendix C, F, H & I). These requirements change mechanisms both share the attribute of having problems with the software and the resolutions to address them take various approaches. M6 captures the situation in which a problem with the software leads to issues in the business problem space. The problem is felt at a different level that challenges the
choice made for a solution. The software solution becomes a problem. For example, the software is not suitable for its purposes (M6, Appendix H: Case A) or a quality attribute is not appropriate, i.e. learnability (M6, Appendix H: Case B) and these raise difficulty for the proponents and end-users. As a result, new needs arise, goals have to be redefined, expectations need to be adjusted and purposes are shifted. Specifically, M6 is an impact relation expected to be found in breakdown situations. However, M6 can also be initially triggered by an initiative which later leads to breakdowns (M6, Appendix H: Case D Items 21 through 24).

The occurrences of M4 impact relations also provide typical instances of breakdown situations due to problems with the software. However, unlike M6, M4 depicts how problems with the system can be circumvented by adapting business procedures, organizational policies and individual attitude about one’s work while retaining the business purpose (Cases B and D). On the other hand, M4 shows also that not in all instances the software has a problem. The software can be a result of an opportunity – an initiative, and comparable workaround efforts have to be taken to promote software use. This is the situation with Case C in which software is already in place and implemented. The concerned department only has to follow-suit (Case C, Appendix F: Item 7).

Requirements statements that are related by the M1 impact relation are articulations of problem situations in which stakeholders feel the need for an improvement. This can take place at the top business level and also at the individual level (Appendix C). The business problem articulation and resolution in the cases involve stakeholders at various levels. Business problems as goals and statements of desires can be opportunistic such as pioneering a knowledge management system and educational innovation (Appendix C: Item 1 and 9). In Case D, the training department tried to incorporate top management concerns in its own goals such as the cutting down costs and in the initial phases, the motivation to take advantage of the growing business opportunities. The same can be said of about M7: Business problem resolved by (re)defining software product concept which is motivated largely by breakdowns (9.6.4). Basically, it is all about how people think they can resolve their concerns either choosing directly a software product as a solution.

Breakdowns are present also in impact relations such as M2, M3 and M5. While M2 and M3 are largely initiative-driven, such impact relations can also be provoked by breakdowns (see Section 9.6.3). For example, the decision to retain use of an imperfect software prompts rethinking about how it can be repositioned in the organization and its anticipated future (Appendix D: Item 13). Likewise, a software product concept can be a mental model that needs correction to enhance software usefulness (Appendix E: Item 27).

9.7.2 Typical initiatives

Typical initiatives are expressed in requirements instances representing the impact relation M5: Business solution leading to improved software solution specifications, and to some extent the impact relations M2: Business solution motivating a software product concept, and M3: Software product concept realized by software solution specification.

M5 is a way of translating business solutions in concrete functions that can be immediately supported by software. It is a mechanism of elaborating the business solution specification into workable software functional and quality units. For example: Case A – the policy of promoting knowledge management in order to facilitate community building provides specific tasks for software to perform such as enabling users to communicate and upload data to share with colleagues; Case B – A policy that requires users to learn the software in two weeks demands that the software has to be learnable within two weeks.

The impact relation M2 is similar to M5 in a way that the software domain is involved to support the business solution but in a less refined way. In M2, the software is not clearly specified
except for its general product. Therefore it leaves room as to how the software product is to be realized and what specific tasks it has to support is not yet determined. This can be seen in the following instances from the cases: Case A - knowledge management policy extended by the implementation of a knowledge sharing system; Case B - replacement of an email system by a larger and richer system such as FCC; and Case C - the course management system as used in the classroom is thought to be extendable as a mobile application.

The translation of software product concepts into working functions and quality attributes as represented by M3 is to a large extent an initiative-driven process because it is all about concretizing and extending an already chosen solution path. For example: Case A – a knowledge management system for enabling knowledge exchange is realized by having the system store data (M3, Appendix E: Case A); and Case B – the concept of an e-learning software is realized by having a functionality that enables online communication between teachers and students (M3, Appendix E: Case B). On other hand, this impact relationship can also be breakdown driven in the form of correction in order to satisfy an implicit requirement based on an already formulated mental model of the software. For example: Case C – the system is seen as an exclusive system for only students and teachers which mean access is restricted (M3, Appendix E: Case C). It is therefore desired as a functionality to have guest access or at least be able to browse general information such as course offerings and timetable without having a user account.

9.7.3 Part-breakdown, part-initiative

In most instances, an impact relations is a mix of breakdowns and initiatives resolutions. In fact all of the impact resolutions are triggered by a combination of breakdowns and initiatives.

Resolving business problems through business solutions (M1) can be motivated either way: by a breakdown or an initiative. The breakdown is depicted by (i) Case A – low level of software use is addressed by suggesting users to populate the system by uploading materials they gather from seminars; (ii) Case C - For teachers, the use of TeleTOP had the side-effect of students not submitting the assignment on time. This breakdown made some of the teachers revert to an offline mode of assignments submission: by hand. As initiatives, the business problem are resolved by (i) Case A - The decision to implement a knowledge exchange tool is an initiative that is met by an effort to involve future users in designing its data structure and (ii) Case B – The implementation project in this case is motivated by the desire to innovate. The innovation is translated in the form of a new software application that will replace a seemingly deficient communication tool.

A quick look at M5 impact relations in the case study data reveals that this type of requirements mechanism is triggered both by breakdowns and initiatives. A well elaborated business solution statement in terms of specific actions to be taken, i.e. Case D using consultants and interns can lead to improved software solution specifications. When motivated by breakdowns, an instance of M5 impact relation is all about taking actions to accommodate software despite its limitations, i.e. Case C limited disk space issue in the Workspace (Appendix G, Item 5).

9.8 Validity issues: threats to validity

In Chapter 3, we identified challenges to data collection that introduce threats to the validity of our findings. While we tried to address these challenges in the process of collecting data from the cases, there are still issues arise with respect to the validity and generalizability of the derived requirements evolution patterns from case data. We can identify some of these threats as follows:

• informal sources of data to establish requirements: user interviews
• reliance on past data possibly compromised by faulty memory
• determining causality from case data

9.8.1 Informal sources of data to establish requirements: user interviews

The use of informal sources of data to establish requirements, i.e., user interviews, gives rise to the concern that the information gathered is of an arbitrary nature and that we are relying on what people said. That the case study method applied in this research relies heavily on qualitative data gathered from interviews in order to establish the requirements does not mean that user interviews were the only source of data. The requirements were also derived from other sources in written form such as project plans and documentation and rich contextual sources such as observations and software artifact inspection which were also carried out during field visits. These data were compared for discrepancies, and any found were resolved by asking questions of the stakeholders. Care was taken to ensure that the case study design and protocol (Sections 3.3 and 3.3.4) which calls for the use of several data sources was followed. What specific sources and methods were used to gather data were documented in the individual case study reports.

Secondly, to establish requirements, the ideal situation is of course to gather them from explicit, written specifications. However, as we have mentioned in the case reports, the small and local implementations of groupware in the cases did not really follow a formal requirements process. In most instances, the stakeholders involved did not have the experience, capacity and knowledge of writing requirements. Therefore, the closest we can come to identifying the requirements is project documentation, if available, and eventually, user interviews. In the last case (Case D), we were able to gather an RFP (Request For Proposal) and other written documents that would somehow indicate what the requirements were. However, the question remains as to how formal and official the written requirements were. Agreements between stakeholders take place in meeting rooms and in informal communication such as email. The fact that the requirements were made explicit does not mean they were the true and valid requirements because the process of approval or agreement to make them official is needed. In most instances, this process is difficult to document. Therefore, on its own, user interviews are useful sources of requirements because they provide opportunities to articulate tacit requirements as a result of reflection. In an ex-post facto reconstruction, stakeholders had the opportunity to look back and justify the choices they made, which basically reflects the requirements that were implemented.

Finally, it is also important to mention that in the fourth case (Case D, Chapter 8) the requirements were derived from a rich data source because the researcher was an employee of the company that was the subject of the case study. Further, data were gathered in an unobtrusive manner. Unlike in the first three cases, which were conducted as external realities to be observed, the fourth case is an intertwined reality. However, there were no significant differences in the quality of data gathered from the first three cases and from the last. The only major difference is that the quantity of requirements were higher in the last case than in the other cases. However, despite this difference, the conceptual framework was validated in the same way as in the other cases. The requirements domains, the various impact relations, and the resulting patterns of impact relation compositions were sufficiently represented in the case.

9.8.2 Reliance on past data: memory issues

When we began the case studies, the roll out of the groupware application has already taken place. Therefore, the requirements have to be reconstructed from previous events. There are several ways to gather data about events in the past and in our studies. As mentioned in section 9.8.1, our study relied heavily on respondent interviews to gather data about the past. Interviews, especially about events that happened in the past, are prone to memory issues. These issues can lead to:
• existing requirements being forgotten
• non-existing requirements being invented while being thought to be remembered (false positives)
• a requirement instance being attributed to a different requirements domain

Several measures have been taken to minimize the risk of memory issues. One way is to apply the principle of triangulation in the data gathering method. For example, having a representative set of interview respondents that would more or less reflect the different stakeholders involved in the implementation can even out varying interpretations of what happened. This approach was carried out in all cases (See sections 3.3.4, 4.1.3, 5.1.4, 7.1.4 and 8.1.4.). Various stakeholders with different roles in the implementation, and more than one respondent per role were interviewed. Another way is to gather observations at different periods after starting the case study and using the same set of questions and respondents. This is also carried out in the cases although the periods are not far apart from each other. Contact with respondents was only during the duration of the case study. We could conduct another round of interview or run a survey at some later period. However, even this runs the risk of memory issues, as people may have changed their minds or have thought about what they previously said and corrected it.

The ideal approach to avoid the threats of memory issues is to have multiple sources of information such as interviews, documentation and observation gathered over time. While we used this approach, we gathered data only once given the limited availability of the cases. The exception to this time limitation is case study D, in which observations were gathered over a prolonged period because the researcher was part of the organization. In both types of cases, the absence of written documentation remains an issue, which makes the choice of interviewing stakeholders the next most reliable source of information.

9.8.3 Evolution patterns as causality

The requirements evolution patterns are based on what were observed to be recurring combinations of decisions documented by the data in the cases. It is difficult, if not impossible to conclude from research that these patterns imply causal relationships. Firstly, due to the rich nature of the data, it is difficult to isolate independent variables from dependent variables and from one another. Nevertheless, the patterns do give a plausible explanation of certain phenomena that have been observed in other projects as well. Secondly, there is also the dynamics of confluence in the sense that a requirement statement in one domain can also be dependent variable given that it is the result of another requirement. At the same time, the same requirement can also impact another requirement domain, thus making the former an independent variable. On the basis of these arguments, it makes sense to refer to these patterns as social mechanisms (Elster 1998; Hedstrom & Swedberg 1998; 1.3.2), which recognize the element of indeterminacy and yet do provide explanation as to why things happened. Mechanisms do not provide the power to predict the way theory as laws do, but they present frequently and easily recognizable patterns across situations that are triggered under unknown conditions or with indeterminate consequences (Elster 1998).

The patterns are not universally applicable in all implementation cases. They are generalizations with a limited scope of applicability, which are considered to be true in situations represented by the cases. In other words, the patterns can occur. The fact that one pattern did not occur in one project does not that mean that the pattern is invalidated. Given another implementation in different project setting, the pattern could be valid. The context in which requirements evolve, especially in a groupware environment is more of a social setting than a natural one. Therefore, approaches and views coming from the social science disciplines which emphasize the difficulty of coming up with universal covering laws for social structures, are not only applicable but also relevant. The requirements evolution patterns can be used as inputs for further research to look
into other types of implementation projects or software products. These can help reveal more about the conditions under which these patterns occur but will not be able to predict the outcome of a condition.

9.8.4 Validity concerns: requirements evolution patterns as plausible change mechanisms

Despite these challenges, the derived requirements evolution patterns are still plausible change mechanisms, mainly because the patterns were observed in multiple cases independent of each other. There is no evidence of a particular bias, which also means that the patterns are robust under small changes in case study data.

Secondly, the patterns represent mechanisms of requirements evolution. The relations and the patterns were derived bottom-up while we were in the process of mapping requirements data into the domains. The patterns also became apparent when combining the data from the case studies altogether. It is not the case that when we formulated the framework that we already identified impact relations that can occur. We let the data fit into the possible domain interaction combinations. In other words, we have not actively searched for data that would give us precisely these patterns. Such an approach would increase the risk of false positives in interviews that rely on people's memories.

Thirdly, the patterns describe change mechanisms that are not unique to the case studies. Once recognized and described, they seem quite familiar because they happen in many projects. Therefore, the patterns are valid generalizations. If we accept the patterns as accurate, the underlying mechanisms consisting of compositions of impact relations provide credible explanations. The claim is not that these patterns would occur in every groupware implementation but rather that these patterns are not unique to the case studies, and have occurred in other projects in the past and will occur in the future as well.

9.9 Conclusion

This chapter is an analytical structuring of multiple case study data according to the requirements evolution framework. The process we took in combining and analyzing the data not only confirmed the usefulness of our proposed framework but it also delivered meaningful and interesting results, namely, requirements evolution patterns. We also identified threats and challenges to the validity of the patterns and provided explanations on how we tried to address these (9.8). The patterns are generalizations that identify recognizable mechanisms in a system implementation. They strongly appeal to our intuitive understanding of software implementation dynamics which to our knowledge have not been described this way before. Therefore, we can say that we have achieved the goals set out in this chapter.

• an overview of the requirements evolution patterns
• a discussion on the usefulness of the conceptual framework and the theoretical implications of the requirements evolution patterns

9.9.1 Summary of requirements evolution patterns

From the discussion of within-domain evolution to impact relations, we were able to identify the patterns of change:

**Refinement process (see Sections 9.4.5, 9.4.2, and 9.4.4.)**

The refinement process pattern is when the resolution takes place in the same requirements domain. This is observed in the continuous evolution of the business solution and software solution specification domains over time (see Sections 9.3.2 and 9.3.4).
Business problem expansion (M1) (See Section 9.5.1.)

-> Based on the M1 impact relation, this pattern is initial ‘no-software’ problem situation for which solutions are sought.

Software problem expansion (M1) (See Section 9.5.1.)

-> An iteration of the M1 impact relation, this is pattern in which a software implementation project generates new business problems for which non-software means are formulated as resolution, i.e. organizational change.

Software product (re)definition (M2) (See Section 9.5.2.)

-> Software product definition is identifying a software product concept that would extend and augment a business solution specification. When the same process is repeated for the same product some time later, it becomes a redefinition process.

Classical specification (M3) (See Section 9.5.3.)

-> From the software product concept, functional, quality and other software attributes that would bring to effect the product idea are derived

Workaround (M4) (See Section 9.5.4.)

-> Workarounds are efforts to overcome the imperfections of a software solution by making changes in the user environment to accommodate the software.

Business solution elaboration (M5) (See Section 9.5.5.)

-> Business solution specifications improve the formulation of software functions and quality attributes.

The software solution becomes the problem (M6) (See Section 9.5.6.)

-> This pattern is a constant reminder that software is imperfect; it does not offer a quick and complete solution and breakdowns are expected to happen along the way.

Software induced business goals (M7) (See Section 9.5.7.)

-> This is the general tendency to jump from problem definition to software product identification.

Top down alignment: M1→M2→M3 (See Section 9.7.1.)

-> Top down alignment is similar to classical linear top-down approach to implementation.

Solution centered alignment: M7→M3→M4 (See Section 9.7.2.)

-> This is the further expansion of the software induced business goals that lead to specifications. It captures the impulse: “We have a problem and this software can solve it for us”.

Opportunistic product recycling: M4→M2 (See Section 9.7.3.)

This is all about re-assigning a new software product concept to the same application without architecting a functional change.

Progressive solution expansion: M1→M5 (See Section 9.7.4.)

Through creative business solution approaches, business problems find resolutions through specific functional and quality specifications of software.

Re-orientation: M6→M7 (See Section 9.7.5.)

When the software solution becomes the problem, it triggers rethinking about the software. Ultimately, this leads to software product concept re-orientation.

9.9.2 Discussion

The requirements evolution framework (see Section 9.3) used to structure the presentation of findings in this chapter and resulting requirements evolution patterns (see Sections 9.4.5, 9.5, 9.6, and 9.9.1) make up the two core theoretical contributions of this research. Relating these to the broader body of knowledge relevant to requirements evolution, two levels of discussion are warranted. Firstly, a reflection on the usefulness and relevance of our requirements evolution framework. How has it helped this research move forward? Secondly, a structured exploration of how the requirements evolution patterns and impact relations relate to existing theories and frameworks. Does it conform to what we know about requirements engineering, software development and groupware implementation? Which new insights have we gained?

Requirements evolution conceptual framework

The requirements evolution framework proved to be a useful analytical tool for structuring and aggregating the findings from the individual cases. The basic set of definitions and concepts served as building blocks for discovering additional concepts: impact relations, breakdowns, initiatives and eventually, requirements evolution patterns. It offers an integrated view of requirements. Below, we will compare it with other integrated views that have been proposed in the literature (Hull et al. 2005; Lauesen 2002; Henderson & Venkatraman 1993; Wieringa 2003; Lankhorst 2005; van Eck et al. 2004).

Integrating the business context and software is inevitable in the current practice of ICT project acquisitions. Multiple parties representing various domains and interests are involved. From the supplier's side, there are sales agents, account managers, technical consultants and project managers who come into contact with various levels of hierarchy. From the buyer's side, there are managers ranging from executive management, project managers, to technical line managers, the IT department and eventually the end-users and support staff. Before a software license agreement is signed, communications at different levels take place and project documentations are written. Requirements, therefore, have a much broader scope than mere software specifications. Business goals need not necessarily come up as part of the software specification document, but they do provide the programmer with the context for the application of function he or she is developing (Lauesen 2002).

Likewise, the distinction between problem specification and solution specification has long been proposed when it comes to specifying requirements (Hull et al. 2005, Lauesen 2002; Kovitz 1999). The formulation of the conceptual framework in Section 2.7 integrates these ideas.

The conceptual framework has been useful in deriving requirements statements which form the basic unit of data for our research (see Chapters 4,5,7, and 8). With each application of the
framework, incremental improvements took place, culminating in the requirements evolution patterns introduced in this chapter. The framework has helped to clarify the findings into these patterns.

**Impact relations: implementation as continuing cycle of problem and solution interactions**

The main novelty in the framework is the treatment of impact relations as objects of study. This allowed to aggregate the case study findings into requirements evolution patterns. The patterns themselves are not new in the sense that they disclose unknown patterns of behavior. In fact they look awkwardly familiar But the gain here is that the mechanics of these patterns have been revealed very clearly, by describing them as composite impact relations. This helps to deepen our understanding of how requirements evolve.

Generally speaking, impact relations, which according to our framework underlie the dynamics of evolution, are linkages between domains. Given that each domain belongs to a more abstract category of problems and solutions, the impact relations are interactions between problems and solutions. Software implementation as a dynamic, temporal process can be regarded as a continuing routine of problem resolution. By implementing groupware, organizations seek to bridge the gap between their current situation and the desired situation. Eventually, in the course of implementation as users carry out their daily work routine, breakdowns happen. Either the system is too difficult to use, it requires extra work, it does not match the work practices of the team and its culture, or its functions and features are vague and poorly understood and users resist the system (Grudin 1988; Ciborra 1996, 2002; Orlikowski 1992; Karsten & Jones 1998; Kumareja et al. 2003; de Vreede et al. 2003; Fjermestad 2004; Kelly & Jones 2001; Brown et al. 2004; see also Chapters 4-5, 7-8). When these issues arise during implementation, the software sponsors usually do something to improve the situation. They introduce training (Case B and D), create documentations, develop workarounds, seek help from ICT experts usually the in-house system administration team or contact the supplier. Somehow, breakdowns and the rise of potential issues are anticipated. In cases B and C (Chapters 5 & 7), each implementation has a dedicated help desk team to support the users. In Case A, the system administration team was closely involved in the implementation. In Case D, the team manager advocates contacting and consulting the LMS supplier. Not only do user organization make preparations for software roll-out, software development companies also anticipate issues and problems for which support and maintenance teams become a standard feature of their organizational structure.

As the impact relations also show, the problem and solution sequence is not necessarily linear in the sense that problems always serve as the starting point and then followed by solutions. It is also possible that solutions lead to still more problems, as the pattern the software-solution-becomes-the-problem (impact relation M6) shows. More than half of the patterns are about the the dynamic interaction between problems and solutions (see impact relations M1, M2, M3, M6). The cycles of problem and solution interactions in various impact relations, specifically the non-linear sequence of solutions leading to problems, provide additional support to the idea of iterative methodologies and development (Agile Alliance 2010; Pressman & Ince 2000; Sommerville 2007). Requirements continue to evolve as the implementation unfolds with new problems being countered and solutions being sought. Support and maintenance form part of the core development cycle. Through these observations, our impact relations further reinforce the idea of requirements engineering as a continuing process (see 2.3.3).

**Requirements evolution patterns and alignment perspectives**

The requirements evolution framework and the impact relation patterns bring to attention the strategic alignment models of Henderson & Venkatraman (1993). The seemingly familiar
geometric sequences of concepts such as business and software between our model and theirs give rise to the question, how does the requirements evolution framework and patterns relate to the strategic alignment model (Henderson & Venkatraman 1993)?

We reckon that there are shared conceptual foundations between our framework and that of strategic alignment. One of these is the inclusion and integration of business and software domains. On the other hand, whereas our model adds the dimensions of problems and solution to the business and software domains, the strategic alignment model demarcates between the external and internal dimensions of business and software. The latter lends to the strategic orientation of the Henderson & Venkatraman model (1993) since it focuses on the economic fit between market and product or service delivery. This is one place in which our model departs from theirs because we focus on the operational aspects, more than strategic, given that groupware implementation is embedded in the internal and functional domain of the organization. Our patterns depict scenarios of software use rather than a broad agenda that addresses the entire organizational structure and its entire IT infrastructure.

Henderson & Venkatraman (1993) offers 4 alignment perspectives for leveraging ICT in the business: (i) strategy execution - a top-down hierarchic process with the business strategy as the main driver and IT is regarded as a support mechanism (ii) technology transformation - implementing the business strategy through an appropriate IT strategy and its articulation into infrastructure and processes (iii) competitive potential - using IT as competitive advantage by exploiting key capabilities in developing products and services to market, thereby subordinating the business strategy to the IT strategy and (iv) service level alignment – building an IT service organization in which the IT strategy and its translation into internal components prescribe how the administrative structure of the organization should be set up. Next to having business and software concepts as common conceptual framework components, we also find similarities with some of these strategic alignment perspectives and our requirements evolution patterns.

Very close association can be found between the impact relation pattern M1→M2→M3 classical top-down alignment (9.6.1) and the strategy execution alignment perspective representing strategic management. Both refer to the hierarchic processes of translating global business goals into software choice. On the other hand, the pattern M7→M3→M4 solution-centered alignment is akin to both the competitive potential alignment and technology transformation alignment perspectives because of the pivotal role of software choice that influences both software and organizational design. Our pattern combines the difference in starting points between these two strategic alignment models. Technology transformation alignment is business driven and leads to changes in the software and its management whereas competitive alignment is software strategy driven leading to transformations in the internal organization. In our pattern, the business problem and the software concept are closely intertwined, in which the thinking about software helps in structuring the business problem. Therefore, whether the alignment is business driven or ICT strategy driven will depend on the context in which it takes place. Case B and D provide good examples of two differing starting points but falling into the same impact relation pattern. Both cases have implemented a similar class of groupware application in the form of learning management system. Both cases also explored the concept of e-learning before coming up with a concrete software package suitable for e-learning such as learning management systems. Case B is an educational institution (and so is Case C, for which this observation is also applicable) who is aware of what is happening in the ICT market in terms of e-learning software. Therefore, we can say that in this case, strategic alignment model of competitive alignment is relevant. On the other hand, Case D also implemented the learning management system as a means to support the newly formed training department. This way, software choice served to support the business strategy, which the technology transformation alignment perspective is about. However, since our model does not
distinguish between external and internal strategies, it is possible for us to capture both efforts in one pattern.

With the exception of the strategy execution alignment perspective and the top-down alignment pattern, the other patterns and strategic alignment perspectives are related to each other nominally by the shared views and emphases on the role of software in the change process. In several instances, the requirements evolution patterns provide alternative views to alignment in a way that system implementation is not always strategic or that it always lead to productive alignment. The patterns re-orientation (\(M_6\rightarrow M_7\)), and the software becomes the problem (\(M_6\)) are examples of alternative alignment mechanisms. The strategic alignment models formulated by Henderson & Venkatraman (1993) provide ideal mechanisms, something that ought to be done in order to achieve value from IT. The requirements evolution patterns on the other hand offer descriptions on how IT implementation takes place in practice, which in terms of groupware as the cases have shown, is not always strategically aligned.

This brings our discussion towards alternative views of alignment that focus on the social dynamics of ICT implementation, on ordinary and informal practices instead of the strategic, and on understanding organizational change as it happens. We will relate the requirements evolution patterns with concepts such as drift, affordances, improvisation and emergence.

Drifting technology

Ciborra (1996; 1997; 2002) used the term drift to refer to the peculiar property of technology, groupware applications in particular, when put into use. Technology drift is “slight or sometimes significant shift of the role and function in concrete situations of usage, compared to the planned, predefined and assigned objectives and requirements that technology is called upon to perform, irrespective of who plans or defines them, whether they are users, sponsors, specialists vendors or consultants.”(Ciborra 2002). Drift captures the messy, unexpected, irregular and unfinished process of system implementation. It is an intrinsic property of groupware implementation which is highly variable and contextual (Ciborra 1996; 2002; Mark & Poltrock 2004), sensitive to user resistance (Orlikowski 1993) but can also be innovative and can lead to increases in productivity (Gunter 1999; Brown 2000), and emergent with shifting purposes (Ngwenyama 1998).

The phenomenon of technology drift is captured in evolution patterns such as software product concept refinement (9.4.4 & 9.4.5) opportunistic product recycling (\(M_4\rightarrow M_2\)), software product redefinition (\(M_2\)), the software solution becomes the problem (\(M_6\)), and re-orientation (\(M_6\rightarrow M_7\)). The software product concept refinement pattern captures technology drift through the continuously changing product identity being assigned to the software without enacting changes to the software itself. In the opportunistic product recycling pattern, we have observed expansions of mental models as users get to know the system and think of other ways on how to use the application. Case D provides a good of example of this in which the learning management system (LMS) also became a tool to support marketing and account management tasks. As part of the opportunistic product recycling pattern, software product redefinition is about the decision originating from the users to shift the system's purpose. The software-solution-becomes-the-problem pattern illustrates drift in which the technology seems to be out of control (Ciborra 1996). We observed this pattern at work in case A in which the knowledge sharing system project failed. The intended users did not support the system because it did not match their way of working and stopped using it. This created a problem for the system sponsors (different from intended users) who tried various means to promote the system. The efforts to steer the direction of the implementation towards a more favorable outcome is also a form of technology drift captured by the pattern re-orientation.

Technology drift and requirements evolution are complementary conceptual formulations that describe the same phenomenon: continuing change, unexpected outcomes and deviations from
plans versus executed actions when it comes to system implementation and use.

**Affordances**

In line with technology drift is the concept of affordances (Ciborra 2002; Norman 1999; Gibson 1979). A software application can have unexpected outcomes when put into use, its role and purpose can shift due to affordances. Affordances are properties of artefacts which people make use of according to how they perceive and interpret these. An artefact’s property suggests ways in which it can be interacted with (Norman 1999). For example, a computer chair with wheels on it rolls. It enables the office worker to move and grab things beyond her reach while remaining seated. As the computer chair provides support and movement, it can also be used to transport things such as a computer monitor when one moves office one room to the other. In this example, the computer chair affords more than sitting. Through affordances, artefacts such as software can be used for multiple purposes aside from what it is intended for.

As open-ended, general-purpose software for supporting cooperative processes (2.4.4), groupware applications are examples of technology endowed with affordances. The cases demonstrate the affordances – instrumentality-in-context of groupware tools. For example, Cases B, C, and D are about learning management systems (LMS) implementations in various contexts – in educational institutions (Cases B and C) and in a software development company (Case D). In each of these contexts, the LMS is used in different ways and purposes according to how the users see fit. In Case B, the affordances of the LMS as a shared communication platform supported the need for a logistics infrastructure that would bring together distributed teacher-trainers who work at home. In Case C, the LMS is used to augment instruction outside the classroom for design students. The LMS application in Case D is used to provide commercial training services and to train new employees.

Affordances are captured in requirements evolution patterns such as:

- software product (re)definition (M2): the properties of the desired business solution give rise to a particular software product concept
- opportunistic product recycling (M4→M2): using the software enables an understanding of what it can do and therefore can think of other ways on how to recycle the system.
- software induced business goals (M7): an understanding of the problem is related to the global properties of a possible software solution.
- re-orientation (M6→M7): software use does not always lead to expected outcomes, a circumspection of its properties can generate ideas for alternative uses when the intended purpose is not met.
- Business solution elaboration (M5): an understanding of work is done, what processes and policies are involved can be translated into functional software properties
- Progressive solution expansion (M1→M5): an expansion of business solution elaboration (M5) with an expanded understanding of the problem can be interpreted into functional software specifications

**Improvization**

Improvization is depicted in various patterns such as workarounds (M4), software induced business goals (M7), opportunistic product recycling (M4→M2), solution-centered alignment (M7→M3→M4) and re-orientation (M6→M7). These are patterns that either go against the flow or skip a process in the classical, linear, waterfall approach to development. By proceeding in a
non-standard, non-linear process, these patterns capture the responses people make to situations and conditions as they take place (Orlikowski & Hofman 1997). Improvisation is situated action taken to cope with unexpected consequences and emergence such that the goal or desired outcomes are still met although with some compromises (Ciborra 1992; 2002). Improvisation can be contrasted with planned and scripted action. Instead of carrying out procedures according to plan, but because the situation has changed or has gone contrary to expectations, one takes an alternative routine of action such that the problem or issue at hand can be resolved. We take the pattern workarounds (M4) as an example of improvisation. The pattern tells us that when users discover that the software is quite limited in providing the appropriate solution for their work, they devise, invent or try to work around the limitations of the tool. They make adjustments in their work processes, circumvent the software by doing things in a different way or precisely use the software in other ways than prescribed, make agreements within the team and create policies of use and document them (see Appendix F).

**Emergent properties**

Groupware implementation in organizations are known to have emergent properties (Ngwenyama 1998; Orlikowski & Robey 1991; Orlikowski 1992; Ciborra 1996, 2002; Orlikowski & Hofman 1997). Organizational changes take place not as a result of uni-directional change but due to dynamic mutually shaping interactions between the software and the organization. The emergent properties are those outcomes that are neither anticipated nor planned. While in some instances it is possible to anticipate some changes to take place, most of the times they are opportunistic and extemporaneous – arising from local innovation and are highly contextual (Orlikowski & Hofman 1997).

The requirements evolution patterns reflect these emergent properties in various ways. For example, the continuous evolution of the software product concept captured by the refinement pattern (9.4) entails the changing view of the system's purpose in the organization are changes that emerge as a result of appreciating the system's capabilities. The emergent properties of the software are hinted by its evolving product identity, which the users came up with because of their improved understanding and appreciation of groupware technology. Likewise, workarounds (M4) and the software solution becomes the problem (M6) are patterns that depict different forms of emergence. Workarounds (M4) are changes to enacted business policies, tasks, processes and individual way of working in order to support software adoption. In an ideal situation, the business solution should actually inform the software solution specification domain through a defined software product concept. On the other hand, this reversed flow of change can be seen as feedback, where the software already being implemented brings about changes to the user environment, and the cycle of feedback goes on. In this manner, one speaks of a co-evolution of problem and solution domains (Dorst & Cross 2001). The pattern M6 represents the regularity of having solutions such as groupware, or for that matter, other ICT tools, when implemented and used, can become a source of problems. M6 captures this phenomenon in the both successful and non-successful implementations. As an emergent property, the pattern M6 depicts the undesired but non-intentional side effects of functional services a software tool can offer.

**Practical software engineering implications**

In practical applications areas such as software engineering, the requirements evolution patterns confirm and support the ideas of old and new software development approaches. For example, the classical top-down alignment M1→M2→M3 (9.6.1) coincides with textbook prescriptions of a linear approach to software development and system implementation such as the waterfall model (Royce 1970; Boehm 1976; Pressman & Ince 2000). On the other hand, it is also generally known that the linear, waterfall process is not necessarily followed in practice. It may be true for
the first iteration but as the implementation progresses over time, the process can iterate into other sequences not necessarily in the same order as the top-down approach. For this reason, new methodologies, mindsets and software engineering approaches have been proposed and developed (Pressman & Ince 2000; Pfleeger 1999; DeGrace & Stahl 1990; Schwabber 2004; Sutherland 2004). Aside from the top-down alignment pattern, the other requirements evolution pattern with their recurring features of non-linearity and iteration support the relevance of post-waterfall methodologies such as the spiral development model (Boehm 1986), Rational Unified Process (Kruchten 2004), agile development (Agile Alliance 2010) and various requirements engineering techniques (Robertson & Robertson 2006; Lauesen 2002).

The non-linear requirements evolution patterns offer plausible implementation scenarios from which software risks can be identified (Boehm 1991). For example, the software-solution-becomes-the-problem (M6), re-orientation (M6→M7) and workarounds (M4) patterns are sources of risk items ranging from having the wrong function and properties to real-time performance shortfalls. Having an insight into the possible implementation scenarios just as the patterns capture, consultants, project managers can carry out the appropriate risk assessment and management strategies for their projects.

**IT support and service management**

Aside from software development, the requirements evolution patterns are also relevant to IT support and service management. As change mechanisms resulting from the implementation and use of groupware applications, the patterns increase awareness about the possible directions of an implementation. These can be of benefit to IT departments who are responsible for delivery, maintenance and support of software systems implemented in their organization or customers. In an informal conversation with an IT support engineer, he said that he deals everyday with workarounds. Knowing that workarounds are regular features of an implementation, IT departments can gain from documenting the alternative procedures and sharing them with users. Cross-functional cooperation between end-users, IT service departments and line managers enhances software comprehension and knowledge integration (Rondeau et al. 2006; Mohan et al. 2008)

**Requirements engineering implications**

As this study is about requirements, the question that arise in this discussion is how do our framework and the patterns relate to the requirements engineering domain (Nuseibeh & Easterbrook 2000; http://www.requirements-engineering.org). For this discussion, we'd like to provide a global overview of what our findings offer to the domain:

**Broader understanding of requirements evolution as it relates to software implementation and use**

The requirements evolution patterns offer views of change mechanisms that take place when software is implemented and used. Requirements evolution in literature (1.2.3; 1.2.4; 2.3) has generally been addressed in terms of changes to specifications and regarded as a regular phenomenon in software development. In our framework, these approaches address only the solution dimension of requirements. Our framework and results take a broader perspective by relating requirements evolution with organizational change. This way, we relate requirements with the changing problem domain. In addition, the requirements evolution patterns also take into consideration the dynamic nature of requirements change. The patterns identify sequences of events and decisions that enable one to see actions taking place instead of categorized descriptions different types of change.
Relevance of requirements engineering efforts beyond software development

As discussed in our literature review in Chapter 2, the focus of most requirements engineering efforts are in development and design of software. In a temporal sense, they are in the pre-implementation or pre-use phase of software. Our research extends the relevance of requirements to the implementation and use phases of software by showing sequences of events and decisions that lead to changed requirements.

Requirements engineering as a dynamic and continuing process

Our study has shown that requirements continue to change after software has been developed, implemented, and used. The process of 'design' does not stop when the software is packaged and released to the customer. Supporting its implementation and managing its delivery of services remain relevant concerns for requirements engineering although they may be given names such as IT service management or support.

We reserve the final discussion in the next chapter about the specific contributions of our research in requirements engineering and how practitioners in the field can benefit from it.
CHAPTER 10

FINAL REMARKS

The main results of this thesis have been presented in Chapter 9. We gave a final update of the requirements evolution framework in (Chapter 6; Sections 9.4, 9.5, and 9.8.1) and presented a set of requirements evolution patterns in (Section 9.6, 9.8.1). In this chapter we wind up the thesis with a look back on the research goals and questions (Section 10.1), a reflection on the research product (Section 10.2 through 10.3), a list of hypotheses based on our results (Section 10.4), and a brief look at practical implications (Section 10.5) and future work (Section 10.6).

10.1 Review of research goals, questions and contribution

There are three things we want to address in this review: the overall goal of this research project (Section 10.1.1), the various levels of questions raised to guide the gathering and analysis of data (Section 10.1.2) and the contribution to research and theory development (Section 10.1.3).

10.1.1 Research goal: theory development

Recalling from Chapter 1, our research goal is to contribute to the development of theory of requirements evolution (Section 1.3.1). The scope is further narrowed down to requirements evolution based on system implementation and use instead of software design and construction, which have been the primary focus of most studies on requirements evolution. We take the application domain of groupware as an appropriate empirical setting to observe requirements evolution during system use.

Recovering the tasks carried out to achieve this goal, in Chapter 2 we explored the literature and defined the scope of our research interest by finding out what is known about requirements evolution in software design and engineering, and of organizational changes surrounding the implementation of groupware user settings. More concretely, in Chapter 2, we laid the foundations for theory development in the form of a conceptual framework which offers a broad definition of requirements (Sections 2.6 and 2.7). From this definition, an initial definition of requirements evolution is formulated. The conceptual framework is broken down in observable units in Chapter 3 so that it can be applied in case studies. The initial application of the conceptual framework in a case study (Chapter 4) has resulted into an update leading to an active definition of requirements evolution as a dynamic process. This definition is given by the concept of impact relations (Section 4.6). Discovery of additional concepts such as breakdowns and initiatives was the result of a second case study in which the conceptual framework is applied (Chapter 5: Case B). In Chapter 6, we present an interim version of the conceptual framework with a list of established impact relations. In Chapters 7 and 8 (Cases C and D), we conducted two more case studies to further confirm the usefulness of the framework and at the same time to gather additional data on requirements evolution as represented by the impact relations.
Ultimately, the research goal is achieved in the final aggregation of findings in Chapter 9. In this chapter, we summarized the collected data from the cases according to the conceptual framework. A continuous dialogue between the framework and data led to the discovery of requirements evolution patterns out of the impact relations (Section 9.5). These patterns represent change mechanisms that appeal to an intuitive understanding of what takes place in a software implementation project. The conceptual framework with its requirements evolution patterns is the most important contribution of this research to the development of requirement evolution theory. It is important to mention at this stage that our results offer one of many possible theories of requirements evolution (Wan-Kadir & Locopoulos 2009; Zowghi & Gervasi 2003). What we have also observed in Chapter 9 is the functional role of the conceptual framework in deriving further generalizations about the phenomenon, specifically in arriving at the patterns. A further discussion on the framework's usefulness and its validity follows in Section 10.2.

10.1.2 Research questions

Various levels of questions have been raised to address different concerns in this research (Yin 2003; Sections 3.3.4 and 6.4.2). At the highest level are normative questions about policy recommendations (Section 6.4.2). These questions are addressed later in this chapter (10.7). Questions at the research level (level 4) and its corresponding sub-question are discussed as follows.

The original central research question \textit{how do requirements change in an evolutionary process of groupware implementation and use?} in Section 1.3.3 specifies the research question in terms of requirements change mechanisms in groupware implementation and use. This question is further worked out in Section 6.4.2 as: \textit{What are the requirements change mechanisms in groupware implementation and use?} Ultimately, the central research question is finds resolution in Chapter 9 with the list of requirements evolution patterns. The conceptual framework with its requirements definitions and a theory of requirements evolution elaborated in terms of within-domain change, impact relations and impact relation compositions contributed to the resolution of the research question. More importantly, this question has been addressed in parts in the early chapters of the thesis through the sub-questions.

\textit{Sub-question 1: What are the impacts of implementing and using a groupware application on the requirements?} In the first sub-question, the effects of using the groupware application on the requirements are documented in the individual case study reports (Chapters 4, 5, 7, and 8) and are summarized in Chapter 9. The cases show that introducing a groupware application in a designated team brings about changes to the operational environment and therefore, to the requirements. The changes to the requirements are reflected in the quality of the requirements that arise. For example, there is the general observation that requirements that arise after implementation (from early implementation to post-deployment) contain more information, are more explicit and as stated in Section 9.3.1, the requirements become more refined. This is established in Case C in which a software solution specification about document upload capacity is not well defined until design students began uploading large files. The upload capacity has to be adjusted accordingly. As emphasized in the findings, the real requirements in terms of what is actually desired out of the system become more apparent only when it has been implemented and used for a while. Specifically, quality requirements relating to interface design, usability and learnability achieve prominence in the later phase of implementation (Cases B, C and D provide a number of examples). Finally, arguing on the basis of the requirements domains, implementing groupware application for use leads to interaction among requirements domains in which requirements instances update each other.

\textit{Sub-question 2: What kinds of situations in groupware use promote requirements change?} This question inquires about situations that promote requirements change. These situations which
mbed an issue have been identified as breakdowns and initiatives (Sections 5.7.1, 6.3, and 9.7). Breakdowns are circumstances of difficulties that need to be resolved and can be expressed as requirements. Case A refers to the mergers between competing companies that could possibility strain working relationships between new colleagues working on the same product line; therefore the larger organization set the goal of community-building. A user in Case B complains about default interface of FirstClass which is filled with unnecessary icons. Students in Case C could not upload their projects in the shared workspace of the course management system. Initiatives on the other hand are actions and to a milder extent suggestions that aim to improve a non-breakdown situation. Embarking new ventures, innovating, thinking of ways on how to improve a seemingly stable working situation are examples of initiatives that can be found in various specific instances in all the cases. Altogether, these two types of situations prompt requirements to change and this have led us to offer the idea that requirements evolution is all about bringing resolution to a situation marred by a breakdown or infused with an initiative.

Sub-question 3: What mechanisms of change describe requirements evolution? The mechanisms of requirements change asked in sub-question 3 have been indicated multiple times as requirements domain evolution, and as impact relations (Chapter 4) and more significantly and intuitively as requirements evolution patterns (Chapter 9). This question is properly addressed in Chapter 9 (Sections 9.4 through 9.6) in which these change mechanisms are referred to as requirements evolution patterns. Further adding value to these patterns are their transformation into intuitive and recognizable labels pointing to common system implementation events and actions.

10.1.3 Contribution to research and theory development

In 9.8.2, we identified the impact relations and its composition into sequences which altogether bring about the requirements evolution patterns as the most important contribution of this research. These ideas shed light on mechanics of change that implicate requirements when software systems such as groupware are put into use. The novelty of this contribution and this work finds support in Mylopoulos (2009) who shared the same observation as ours:

Yet, evolution of software system species has been studied only at the level of code and design, but not at the level of requirements. In particular, there has been considerable research on software evolution focusing on code re-engineering and migration, architectural evolution, software re-factoring, data migration and integration. However, the problem of post-deployment evolution of requirements (as opposed to architecture, design and/or code) has not entered into the research discourse.

10.2 Research product: conceptual framework

The research product is a conceptual framework (Section 6.1; Figures 6-1 and 6.2; Table 6-1) that offers a view of evolution as a dynamic temporal process constituted by changes in requirements domains. It is made up of:

• definitions and constructs of requirements as requirements domains (Sections 2.7.1 and 3.3.1; Figure 2-5).
• a description of requirements evolution as a change process captured by changes in requirements domain (Sections 2.7.23.1.1, 6.1, and 6.2; Figure 2-6).
• a classification of different requirements change mechanisms as impact relations (Sections 4.6, 5.6, 6.1, and 6.2; Figures 6-1 and 6.2; Table 6-1).
• concepts such as breakdowns and initiatives to explain the underlying mechanisms of requirements evolution (Sections 5.7.1 and 6.3).
• a simplified explanation of requirements evolution mechanisms as patterns depicting familiar
and recognizable features of system implementation projects (Section 9.8.1).

In Section 1.3.2, we referred to the research product as theory. There are many conceptions of what theory is depending the level of abstraction and the discipline involved (Silverman 2000; Wieringa 2009). Our conceptual framework is consistent with the following formulations of what theory is:

• an arrangement of a set of concepts to define and explain some phenomena; a basis for knowing how what is unknown might be organized (Silverman 2000)
• a theory-of-many where \( n \leftrightarrow k \), which is generalizable only to specific classes of problems, conditions and phenomena (Wieringa 2009)
• as statements for analysis and explanation. It states the phenomenon as it is, how, why and where but does not aim to predict with precision (Gregor 2006)
• as a trick that shows a way around some common difficulty by suggesting a procedure that resolves relatively easily what would otherwise be an intractable and persistent problem (Becker 1998)

With this, we distinguish our conceptual framework from other perspectives on theory as testable universal statements (Popper 1980), grand or formal theory (Bachrach 1989), or prescriptions for action, as in a theory of design (Gregor 2006; Markus et al. 2002). We do not categorically say that our conceptual framework is devoid of prescription given the nature of the problem and the knowledge domain it aims to contribute to. It makes more sense to say that our framework has practical implications for design and we can provide a few recommendations.

Going back to our bulleted list of theory definitions, our conceptual framework is a system of concepts including their plausible relations, which further refine abstract artefacts such as requirements and phenomenon such as requirements change (Silverman 2000). The relations are hinted by the introduction of concepts such as impact relations, which connect one requirements domain to another, thereby providing an idea of a mechanism that underlie requirements change (Gregor 2006). These relations help us address the question of how. We also refer to a theory-of-many when we speak of the scope and applicability of the conceptual framework (Wieringa 2009; 2010). When we try to make sense of requirements evolution using our framework, we basically refer to the impact of a particular class of software to its environment i.e. applications that support non-structured social processes of a group working together such as groupware. Embedded software systems and other ICT applications that do not interact with human agents are not expected to behave according to our requirements evolution patterns. As a trick, our conceptual framework guided us generating data from the cases, to seek for observable indicators of requirements and requirements change and to further diagnose the situation (Becker 1998). A conceptual framework is therefore a functional theoretical tool.

Sections 10.2.1 through 10.2.4 discuss and reflect on the main ideas espoused by our conceptual framework.

10.2.1 Requirements domains

Our conceptual framework has as its starting point the requirements domains (Sections 2.7.1 and 3.3.1; Figure 2-5). The domains offer an integrated view of what requirements are. Chapter 2 provides the basis for the creation of this framework. In our view, there is little novelty in the formulation of the domains in the sense that integrating the domains of business, software, problems and solutions have already been suggested in literature. What is more important to mention on the other hand is the role it played in moving the research further. Through this definition, we were able to further build the model. We were able to begin with, start with the gathering of requirements for the first case study (Chapter 4).
10.2.2 Requirements domains as impact relations

The impact relations provide the key concept for understanding requirements evolution (9.5). They supply us with the underlying mechanism of requirements change. With this understanding, derived from empirical observations, we improve the frontier of our knowledge about requirements evolution, which as earlier noted (Mylopoulos 2009; Sections 1.2.4 and 2.3) have been limited to describing changes in written specifications. Through the impact relations, we can know why requirements change. This is a step further than knowing that requirements have changed and classifying requirements evolution types into stable and volatile requirements (Sommerville 1998).

10.2.3 Requirements evolution patterns

The requirements evolution patterns make use of commonly understood terms to refer to the different mechanisms of requirements change (Section 9.4.5, 9-5, and 9.6). The patterns were based on the impact relations and their composite sequences. Therefore, next to the impact relations, the patterns are extras that make the framework more communicable. Ultimately, the requirements evolution patterns are the mechanisms of change expressed in familiar terms. The patterns can be regarded as scenarios of system implementation in which various stakeholders can benefit in knowing.

10.3 Evaluation: usefulness criteria

The quality of conceptual frameworks as theoretical tools can only be evaluated by the extent to which they have been useful. As Silverman (2000) points out: Models, concepts and theories are self-confirming in the sense that they instruct us to look at phenomenon in particular ways. They can never be disproved but only deemed to be more or less useful. In turn, a conceptual framework can be considered useful when (Wieringa 2009):

- it shows structures in reality
- it allows recognition of entities
- it allows communication about entities
- it allows generalization
- it allows prescription
- it allows for making true or false statements

Further along these lines, we can evaluate the usefulness of our conceptual framework.

10.3.1 Structures in reality

Our conceptual framework is an abstraction of objects and phenomenon that can be found in reality, circumscribed in the domain of software development and organizational implementation of ICT. While the objects may be abstract and artificial in nature i.e. requirements, business problems, requirements evolution, they are objects that exist in the conversation and exchange of information among human agents that populate the domain.

Through our definition of requirements and its representation into constructs such as business problems, business solutions, software product concept and software solution specification, we tried to reflect the structure in reality that software requirements are interpreted differently by different stakeholders involved in the process. Further, in the process of constantly updating the framework with empirical data, deriving additional objects such as impact relations to conceptualize requirements evolution, we strive to remain truthful to the domain. Basically, we let the case data define the concepts relevant to understanding requirements evolution. This way, our concepts point to something that exists externally outside the framework. In terms of Becker
(1998) concepts are ways of summarizing data.

10.3.2 Recognition of entities

Section 9.8.2 discusses the pragmatic usefulness of our conceptual framework as a functional tool in gathering research data. Delineating the scope and the definition of requirements using the domains (Sections 2.7.1 and 3.3.1) enabled us to identify requirements from cases. The various compositions of impact relation sequences enabled us to recognize familiar patterns in system implementation,

10.3.3 Communication about entities

The externalization of our conceptual framework as an artifact into statements and diagrams (Section 2.7, 3.3.1 and 6.1-6.2; Figures 2-5, 2-6, 6-1, 6.2, and 9.8.1; Table 6-1) is a vehicle for communicating and visualizing our view of requirements evolution.

10.3.4 Generalization

There are several levels of generalization one can speak of. One is about all about establishing external validity (Yin 1994). This implies that results of one study can be replicated in another study. The other is analytic generalization, which is generalizing to a broader theory (Yin, 1994). To generalize to a theory is to provide evidence that supports that theory (Firestone 1993). In the case of our research, we strive towards analytical generalization. Our results, in particular the patterns, say something about changes that take place during system implementation in organizations. They provide evidence towards the broader theory of emergence in requirements for systems-in-use.

10.3.5 Prescription

According to Eddins (1967), conceptual frameworks can be used as ideological levers and rallying points for action. Section 10.2 identifies the practical implications of our framework given the domain and the nature of the problem it is based upon. After all, gaining an understanding of the problem or the phenomenon very closely implies preparing for and taking action.

10.3.6 True or false statements

In other words, a conceptual framework should lead to the generation of research hypotheses. Whereas conceptual frameworks cannot be validated either as true or false, hypotheses on the other hand can be tested. Section 10.4 provides a list of hypotheses formulated based on the conceptual framework.

10.4 Research hypotheses

A conceptual framework is considered useful when one can generate testable hypotheses out of it (Section 10.2.1). The following are few research hypotheses (Hn) that we were able to draw from our research. They also form as part of our research deliverables.

H1 Requirements evolution need not necessarily lead to software change.

The software can acquire a new product identity while the specification and functionality remain intact. See software product concept (re)definition (Section 9.5.2) opportunistic product recycling and problem-driven product concept evolution (Section 9.6.3) and re-orientation (Section 9.6.4).
H2  Requirements evolution is the resolution of a breakdown or an initiative identified in one
requirements domain and resolved in another requirements domain.

These were be enacted through all of the 7 impact relations and 5 impact relation compositions.

H3  There is a tendency to jump from business problem identification to software product concept
identification.

In other words, business problems are defined, refined and resolved by means of software. See
software induced business goals (M7) pattern.

H4  When the resolution leads to a software decision, the specification is not more than the software
product concept level.

See software product (re)definition (M2) pattern and business solution elaboration (M5) pattern.

H5  Only after using and having implemented the software for some time (i.e. early implementation)
do the requirements pertaining to the software solution specification become apparent and
known.

See classical specification (M3) pattern.

H6  Using and interacting with the software lead to breakdowns.

See workarounds (M4) and the software solution becomes the problem (M6) pattern.

H7  The resolution of a breakdown does not always have to result to action for evolution take place.
A resolution can be resolved by non-action.

See Case B and Case C (M4, Appendix F: item 6 & item ).

H8  In later phases of the implementation, there is the general tendency to justify the existence of the
system.

See pattern re-orientation (M6→M7).

H9  Software's use prompts the software's concept to change.

See software product (re)definition (M2) and opportunistic product recycling (M4→M2).

H10 In case of software solution specification breakdown, one finds a new business problem or a
business solution that the software might solve and support. When these are found, the software
product concept is changed.

See opportunistic product recycling (M4→M2) and re-orientation (M6→M7).

10.5 Practical implications and recommendations

Moving from theoretical to practical implications, the findings of this research and the lessons
learned from the case studies offer several insights and recommendations for action. These have
been anticipated in the following level 5 questions (Sections 3.3.4 and 6.4.2):

L5Q1 On the basis of the study conclusions, what actions or guidelines can be given to organizations
hosting groupware that will help them manage the process more efficiently and purposefully?

L5Q2. What advice or heuristics can be given to designers of collaboration technologies such that the
uptake and use of these technologies are indeed supportive of human activities and purposes?
10.5.1 Guidelines for groupware hosting teams and organizations

L5Q1 is all about improving software use and basically maximizing the benefits from software. As the case studies show, teams implement software solutions such as groupware in order to improve their work processes. They also show that this effort is full of challenges. We have identified the following guidelines and recommendations to organizations and teams hosting as well considering implementing groupware tools.

**Actively involve users in the implementation; make them part of the project if possible**

It has been widely articulated that user involvement in software design and implementation can contribute to its success (Markus 1987; Grudin & Palen 1995; Nies & Pelayo 2010; Henfridsson 2010). While project success can be defined in various degrees, what we can demonstrate on the basis of the case studies is that involving users can help achieve certain operational goals. For example in Case A, the involvement of end-users helped in identifying specific data requirements for the knowledge management system. Users as expert performers of the specialized processes in their work can provide meaningful design inputs. From the user interviews in the case studies, user articulations of system breakdowns and how they use the system are furnish the detail needed in a specification that can be translated as software properties.

The extent to which users are involved in a project also varies. It depends on how much time and resources can be spared for the project. In some instances, a consultative approach is sufficient. Key end-users are approached, their feelings about the software and inputs for design are gathered. What is more important is the process on how the project is carried out, that key users do not get the feeling that they have been left out. This is one of the lesson learned from Case D in which a student intern was tasked with the search for an open source LMS to replace the currently being used learning management system (LMS). Some members of the team felt left out, especially the advanced users. The open source LMS did not push through. For a more involved participation, advanced users can be asked to actively take part.

**Project documentation**

A common feature of the implementation in the case studies is the lack of requirements documentation. Since most implementing teams are not software experts, it is not expected that they carry out a formal requirements process. What is important is that the requirements, the problem and the resources are documented. This helps in communicating the project and preserving it. It also helps in cases of transitions, i.e. when new people join the team or somebody else takes the project over.

**Involvement of experts**

When resources allow, the involvement of experts in the domain in question can help improve the process of requirements specification and selection of the appropriate software. This was observed in Case D in which consultants have been hired to assist in the development of a training curriculum and in the selection of a learning management software. The use of experts compensates for time and the knowledge gaps in the team members.

**Workarounds: training and written updates**

The pattern workaround (M4) is a frequently occurring implementation scenario which can be rather seen as regularity. It is expected to happen. In recognizing that workarounds are expected to happen, training and regular written updates can reduce the learning curve and help users to adapt the software much faster. Documenting workarounds and communicating them help save time and avoid frustrations.
Closer cooperation and partnership with the IT support and the implementing team

The implementation of a software application in an organization or even in a local autonomous team is not without the involvement of the IT support department (in some organizations called system administrators) who manages the IT resources. Groupware is networked application that links people through their computers imply more than just installation in its implementation and use. Therefore, it is important for the hosting team to have close cooperation with the internal IT support department. The latter's support and involvement is needed when it comes to software updates, resolving workarounds and other issues with the software.

Pushing implementation from breakdown

As we have found in this study, requirements are articulations of issues which can be in the form of a breakdown or an initiative. There is benefit in pushing the implementation from a breakdown in the sense that breakdowns appeal more because they diffuse a sense of urgency. Breakdowns magnify the problem being represented. This way, a system implementation whose motivation is largely of an initiative can also be pushed forward by identifying potential breakdowns that can be avoided by the intention. This is does not mean that initiatives are less preferred than breakdowns. Initiatives are needed in order to bring about innovation but such an innovation needed the support of the user environment and top management.

Group first, groupware second

As already mentioned in lessons learned in Case A, one cannot create a group or stimulate group processes by means of implementing groupware tools. Groupware applications are meant to support group processes that exist or at least a group that already exists whose cooperative processes need further stimulation and facilitation.

10.5.2 Software design heuristics

While our research focused on software use, there are also several design guidelines that can be derived from this exercise which can help in the development of groupware technologies (L5Q2). As groupware functionality is becoming ever present in business applications software, the following guidelines can be useful.

Understanding of user tasks especially non-structured processes and ensuring performance

Groupware applications are used for supporting non-structured work processes. These are the processes that are not formally specified but do take place regularly in the course of work. Therefore it is important to also focus on the non-structured processes, i.e. informal communication exchange, group processes, i.e. how meetings are conducted and collaborative activities, i.e. intensive interaction between teams, i.e. writing documents together, working on the same project with different tasks at the same time. What is also important is that in supporting these functions, the performance of the application is not compromised. For example, nothing is more irritating in a video-based conferencing than delays in message transmission that slow down the communication.

User feedback

Usually, software developers especially of COTS applications are external to the host organization and teams. Therefore, they are outside to the day to day operations and use of the software. It is important that from time to that developers get insight into how the software is being used. Some large software companies organize annual user conferences.
Software quality through training, documentation and support

Software accessibility is a well-articulated problem raised in the cases. This underscores the complexity of software systems for which training, documentation and help are needed in order to make it accessible. Documentation and help should be visible and should be written instructively, i.e. what is the starting point, and with a number of examples.

Integration and awareness of related or 'redundant' applications

With the increased integration and presence of software in business today, it is usually the case that the groupware application is just one of the many software tools in the organization. Usually, the hosting team would like to integrate the groupware with the other applications they are using. Or they make use of the groupware in relation with other existing specialized applications (Case C and D). Chances are the groupware shares some redundant functions with existing applications, i.e. Case C: TeleTOP is used together with student portal, course information system, exam registration system. Therefore one speak of an environment, a broader platform of related applications for carrying out a complete organizational process. For example, in Case D, carrying out the training delivery process involve the use of software training lab applications and VMWare images next to the learning management system (LMS). Software developers should take this increasingly common situation in consideration and should find ways on how to uniquely position the groupware application in the implementation.

Avoid redundant function names

In Case C, we have encountered the issue of poor interface design and ambiguity in software functions. For example, the difference between the option 'Presentation' and 'Archives' are not apparent for users. Designers should take into consideration informative function names and should be distinguished from each other in a way that one does not become redundant.

Breakdown focused requirements assessment

Designers can also benefit from utilizing a breakdown perspective in gathering requirements and in introducing innovation. By focusing on breakdowns, attention is directed towards interruptions, items that break down, disengagement and non-transparency such that the necessity of taking action and doing something become urgent.

10.6 Directions for future work

Investigating the topic of requirements evolution for systems-in-use is a promising area for further research. It relates to a broad range of issues both theoretical and practical. Specific to the contributions of this research, there are 5 areas we can identify as directions for future studies.

10.6.1 Methodology: theoretical sampling using different theoretical categories

Our case study design (Section 3.3.3) is based on a theoretical sampling that made use of polar types. The main theoretical category used in selecting cases is the implementation outcome: successful vs failed implementation. Additional case studies can be done to extend the findings by using different theoretical categories such as small vs. large projects, co-located vs. virtual teams and different business domains, i.e. financial vs. education, software vs. manufacturing, etc.

10.6.2 Validation of research hypotheses

Validating the hypotheses formulated in Section 10.4 is another area of future work that can
extend the contributions of this research. These hypotheses can be applied in additional case studies and used for selecting theoretical categories (Section 10.6.1).

10.6.3 Exploratory research on additional patterns

The discovery of several patterns in this research stirs further inquiry into the possible existence of other patterns. The question lingers after the identification of the 7 impact relations (Chapter 6) and impact relation patterns is that: are there additional patterns that can be discovered? For this, additional case studies can also be done and the suggestions from Section 10.6.1 can be taken into consideration.

10.6.4 Usefulness of conceptual framework to other application domains

Finally, an interesting future study direction is the investigation of requirements evolution in another application domain other than groupware using our conceptual framework. To which extent is the model applicable to other software application domains? What patterns can also be found in those domains and are there also other patterns that can be discovered? This endeavor would broaden the reliability and usefulness of the framework. More importantly, it would provide an interesting comparison between different types of software, i.e. between social communication software such as groupware and specialized software with specific commercial and operational purposes such as reservation systems, library information systems, and bookkeeping software.

10.6.5 Design research into requirements engineering for systems-in-use

This study is a theoretical empirical approach in investigating requirements evolution for groupware applications in use. Its goal is to address gaps in knowledge about requirements evolution as a phenomenon leading into the formulation of a conceptual framework and the discovery of evolution patterns. This approach has been contrasted with applied research in which solutions are formulated to address practical problems (Section 1.2.4). However, in order to devise solutions, some knowledge or theory about the problem and the potential solution are needed. In this manner, the contributions of this study can provide inputs for practice-oriented discipline such as requirements engineering. As a potential area for future work, this study finds direction in the line of design-oriented research for requirements engineering for systems-in-use. A venture into this direction finds support in proposals for a continuing requirements engineering process (Jarke & Pohl 1994; Section 1.2.4), software evolution (Lehman & Ramil 2003) and post-deployment requirements (Mylopoulos 2009). This future research direction can further take shape into various topics such as:

- designing and applying a method for post-deployment requirements engineering; and
- prescribing and evaluating design interventions for certain software implementation scenarios i.e. evolution pattern M6. The software solution becomes the problem, such that the project becomes a success, and re-orientation (M6 → M2) with the goal of influencing the pattern and its consequences.
Diagnostic (guide) questions for end-users / managers / support

1. Background information
   • What do you do here? Job title/position, years working for the company, department and group
   • What is your background?

2. Job and task analysis
   • What are your job functions or responsibilities?
   • What are your tasks? Describe your typical activities in a normal working day.
   • What is the most critical aspect of your job
   • What kind of problems do you encounter when you perform these tasks? What aspects of your tasks are so irritating for you? When problems occur, how do you handle them?
   • If there were some things that need to be improved with regards to your task, what would be those?

3. Software usage
   • Do you use the system <<name of groupware application >>?
     – If used:
       • For what purposes do you use the system? For which particular tasks?
       • Which specific functions of the system do you use?
       • Are you completely dependent on the system to do your job?
       • Do you think that the way in which you use the system is the way on how it is intended to be?
       • Do you use the system in some other way that intended?
     – If not: Why don’t you use it? What will make you use it?
   • Does the system provide you with sufficient and precise information to enable you to do your tasks?
   • Are you satisfied with the way the system is functioning for you?
   • Do you find the system easy to learn to operate and use?

Breakdowns and Initiatives

• What kind of problems do you encounter when you use the system? Are there some errors/difficulty in the program that require you to work around it?
• Do you recall a moment when the system was not working and you cannot do your job? What did you do then?
• What kind of limitations do you see in the system? What kind of complaints do you have?
• Are you aware or are there any efforts within the department towards continuously improving the system or the state of the implementation?
• When you have suggestions for changes or improvement (new functionality, new way of how it is to be organized), do you go your way out and put these forward?

4. Cooperation and group processes
   • How is cooperation structured in your group / department? With whom do you have to work inside and outside the group to get things done? How does the communication take place – by phone, informal discussion, meetings, telephone, written report, email, online discussion? Which of these cooperative processes does the current system support?
   • To which extent do you use the system for working together, i.e. making decisions, planning or working together on a project, sharing files and documents, scheduling or agenda management? Do you feel the
need to use the system because your colleagues are using it?
• Do you get encouragement from other colleagues to use the system?
• When you have difficulty using the system, do you contact your fellow colleagues for help? If so, how?
• Do you feel free to use the system (in the sense of not being forced)? OR: Do you feel being limited to use the system from other purposes that you see fit?
• Do you take some time, as a group, to include in your meetings to talk about the system, about your experiences, problems and ideas for improvement?

5. Contextual Factors at the Individual Level
• Do you like working with computers? How long have you been working with computers?
• Do you have previous experience of using group or community support applications such as email, scheduling systems, discussion and knowledge-sharing systems? If so, please provide details.
• Are you aware of the possibilities that the current system can offer? Does by using the system increase your awareness of what you can do with it?

6. Support
• User training: Is there user training given? Do you think the training that was given was enough?
• User satisfaction monitoring: Are there any visible efforts that you can think of that is meant to continuously monitor user satisfaction?
• Does your manager / team leaders give you support in terms of time, resources as well as morale in learning to use, operating and experimenting with the system?

7. System use policies
• Are there any form of policies or agreements regarding how to use and when to use the system? How formal or informal are these?

8. Organization of the RE Process
• Were you personally involved during the decision-making phase in the form of a problem diagnosis / need analysis process leading to the implementation of the system? If yes, can you please narrate your experience? If not, do you think it is important that you are personally involved in decision-making or that your needs be analyzed whenever there is a planned system implementation?
• Was group given the autonomy to make decisions about what system to choose or what functionality you would like the system to have?
• Were you involved in specifying requirements?

9. Changes and evolution
• What changes have occurred ever since the system was implemented, with regards to:
  – The way you communicate with your colleagues?
  – The way you collaborate or work together?
  – Quality of work life?
  – The organizational performance as a whole
• Are you aware of any organizational concerns that you somehow might affect the implementation and use of the system?

Other sources of data checklist
• Project documents
• Software artifact inspection or demonstration
• Software product website
• Company website and reports
### Appendix B.1  Business Problem Domain Requirements Statement

<table>
<thead>
<tr>
<th>Item</th>
<th>Case</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>PRE_BP_1</td>
<td>Active Insurance Group intends to respond to continuing mergers and acquisitions.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>PRE_BP_2</td>
<td>One of the goals of the organization is to build a community of employees and to evoke a feeling of ‘one AIG company’.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>PRE_BP_3</td>
<td>The KCS - non-life insurance aims to translate the organizational goal of integrating and unifying the different sub-companies by developing a community of non-life insurance experts</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>PRE_BP_4</td>
<td>The KCS - non-life insurance is tasked with developing the competency of the non-life insurance experts</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>PRE_BP_5</td>
<td>The non-life insurance experts would like to perform their tasks efficiently and to be able to gather the information they need from colleagues in other division in an uncomplicated manner</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>EARLY_BP_1</td>
<td>KCS wants to have a system that the non-life insurance experts will use.</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>PRE_BP_1</td>
<td>The institute wants to participate in the renewal and modernization efforts in the educational sector through ICT.</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>PRE_BP_2</td>
<td>The institute wants to implement an e-learning software.</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>PRE_BP_3</td>
<td>IT Department and FCC administrators at OI. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>EARLY_BP_1</td>
<td>The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.</td>
</tr>
<tr>
<td>12</td>
<td>B</td>
<td>EARLY_BP_2</td>
<td>The teacher-trainers want to improve their competencies. Specifically, they want to improve their knowledge and skills about FCC.</td>
</tr>
<tr>
<td>13</td>
<td>B</td>
<td>POST_BP_1</td>
<td>The teacher-trainers want to broaden the their useful functions of the software in addition to the ones that they already know.</td>
</tr>
<tr>
<td>14</td>
<td>B</td>
<td>POST_BP_2</td>
<td>Teacher-trainers would like to get to know other colleagues in a more personal manner than online.</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>POST_BP_3</td>
<td>Teachers would like to work more in teams when doing consulting for schools.</td>
</tr>
<tr>
<td>16</td>
<td>B</td>
<td>POST_BP_4</td>
<td>Users get the feeling that work never stops and that there is always a sense of urgency to reply to queries.</td>
</tr>
<tr>
<td>17</td>
<td>B</td>
<td>POST_BP_5</td>
<td>Users do not want to be flooded with unnecessary information on their FCC start-up screen.</td>
</tr>
<tr>
<td>18</td>
<td>C</td>
<td>EARLY_BP_1</td>
<td>Students should submit their assignments on time or ahead of time. The continued availability of TeleTOP blurs the official notions of time. Students submit their assignments up to the last minute of the day; educational processes continue 24/7.</td>
</tr>
<tr>
<td>19</td>
<td>C</td>
<td>EARLY_BP_2</td>
<td>Students should not be able to copy each other’s work or look into other group’s solution.</td>
</tr>
<tr>
<td>20</td>
<td>C</td>
<td>EARLY_BP_3</td>
<td>The Roster is not filled up completely and consistently by teachers.</td>
</tr>
<tr>
<td>21</td>
<td>C</td>
<td>POST_BP_1</td>
<td>Teachers would like to know the number of students who will be taking the course so that they can be distributed beforehand.</td>
</tr>
<tr>
<td>22</td>
<td>C</td>
<td>POST_BP_2</td>
<td>The availability of software applications that provide overlapping services create a sense of discontinuity in the process. Users want an integrated environment that supports the process or they want to deal with only just one application.</td>
</tr>
<tr>
<td>23</td>
<td>D</td>
<td>PRE_BP_1</td>
<td>Aside from consulting services, customers demand for more structured and formal knowledge transfer efforts such as training and documentation regarding the use and implementation of FDS software.</td>
</tr>
<tr>
<td>24</td>
<td>D</td>
<td>PRE_BP_2</td>
<td>For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.</td>
</tr>
<tr>
<td>25</td>
<td>D</td>
<td>PRE_BP_3</td>
<td>The new training department is not experienced with the provision of training services and development of training materials.</td>
</tr>
<tr>
<td>26</td>
<td>D</td>
<td>PRE_BP_4</td>
<td>Training handouts should be available and accessible to trainees during and after the training.</td>
</tr>
<tr>
<td>27</td>
<td>D</td>
<td>PRE_BP_5</td>
<td>Customers would like to have training conducted in their own offices.</td>
</tr>
<tr>
<td>28</td>
<td>D</td>
<td>PRE_BP_6</td>
<td>FDS feels the need to bring (new) employees up to speed in term of knowledge and competency.</td>
</tr>
<tr>
<td>29</td>
<td>D</td>
<td>EARLY_BP_1</td>
<td>FDS partners should also get training on FDS data solutions.</td>
</tr>
<tr>
<td>30</td>
<td>D</td>
<td>EARLY_BP_2</td>
<td>LMS Supplier who responded to the RFP was too expensive.</td>
</tr>
<tr>
<td>31</td>
<td>D</td>
<td>POST_BP_1</td>
<td>Increasingly, Sales and US-based account managers continue to request for onsite training to customers who didn’t sign a software license agreement yet. Expressed knowledge gap by customers on FDS software is seen as a training need.</td>
</tr>
<tr>
<td>32</td>
<td>D</td>
<td>POST_BP_2</td>
<td>Training session set-up is increasingly becoming tete-a-tete, Oracle iLearning performance issues are becoming more prevalent. Downtimes are getting frequent; there is almost no training session setup with no down time or error messages received while uploading new materials.</td>
</tr>
<tr>
<td>33</td>
<td>D</td>
<td>POST_BP_3</td>
<td>Negative course feedback received as trainee access to learning content, i.e. a pdf file or slide is getting slower.</td>
</tr>
<tr>
<td>Item</td>
<td>Case</td>
<td>Code (R)</td>
<td>Requirements statement</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>34</td>
<td>D</td>
<td>POST_BP.4</td>
<td>Quality of training deteriorates due time outs during onsite training. Trainer reports that the course evaluation takes too long to load and often leads to time outs.</td>
</tr>
<tr>
<td>35</td>
<td>D</td>
<td>POST_BP.5</td>
<td>New FDS training manager wants to cut down costs.</td>
</tr>
<tr>
<td>36</td>
<td>D</td>
<td>POST_BP.6</td>
<td>Newer members of the team, i.e. the new training developer finds Oracle iLearning a difficult system to use. Even after having an intensive orientation, she finds training setup tasks in Oracle to be too complicated.</td>
</tr>
<tr>
<td>37</td>
<td>D</td>
<td>POST_BP.7</td>
<td>After so many tries, the new training developer cannot get an assessment up and running in Oracle iLearning.</td>
</tr>
<tr>
<td>38</td>
<td>D</td>
<td>POST_BP.8</td>
<td>User help and how-to tutorials, especially in setting up assessments, seem to be unavailable.</td>
</tr>
<tr>
<td>39</td>
<td>D</td>
<td>POST_BP.9</td>
<td>Due to a lack of clear direction and manager commitment, the intern left the open source LMS project. Oracle iLearning will not be replaced in the meantime.</td>
</tr>
<tr>
<td>40</td>
<td>D</td>
<td>POST_BP.10</td>
<td>FDS is not able to sign any license agreement in the last 6 months of 2008. The prognosis for next year is zero sales. Customers are cancelling maintenance and support agreements.</td>
</tr>
</tbody>
</table>
### Appendix B.2 Business Solution Domain Requirements Statements

<table>
<thead>
<tr>
<th>Item</th>
<th>Case</th>
<th>Code (R.)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>POST_BS_1</td>
<td>The users should be encouraged to make use of the system by organising meetings to talk about the system and finding ways to use it.</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>POST_BS_2</td>
<td>Users should put training and seminar materials in KENNISNET. If not available in electronic format, a brief summary of the training/seminars/courses should be written and be made available for other colleagues through KENNISNET.</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>POST_BS_3</td>
<td>Users should publish in KENNISNET market research and competitor reports that were conducted in their sub-companies</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>POST_BS_4</td>
<td>Users should also publish work methods or processes of how a work problem or project was approached and solved</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>EARLY_BS_1</td>
<td>FCC will replace the currently existing Pegasus email system.</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>EARLY_BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>EARLY_BS_3</td>
<td>All new teacher-trainers at O1 have 14 days to familiarize themselves with FCC.</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>EARLY_BS_4</td>
<td>The IT department should also function as a helpdesk and provide technical assistance to users when they questions or problems with the software.</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>POST_BS_1</td>
<td>All teacher-trainers should begin using FCC in communicating with their students.</td>
</tr>
<tr>
<td>12</td>
<td>B</td>
<td>POST_BS_2</td>
<td>The delivery of instructional materials to students should be in electronic form and disseminated through FCC.</td>
</tr>
<tr>
<td>13</td>
<td>B</td>
<td>POST_BS_3</td>
<td>New employees should be trained on how to use FCC.</td>
</tr>
<tr>
<td>14</td>
<td>B</td>
<td>POST_BS_4</td>
<td>Apart training sessions should be given for advanced system functionalities and features, tips and tricks, as well as proper ways of using certain system functionalities.</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>POST_BS_5</td>
<td>Users should organize and sort information on their screen in order to avoid information clutter.</td>
</tr>
<tr>
<td>16</td>
<td>B</td>
<td>POST_BS_6</td>
<td>Do nothing: continue using FCC despite the unpleasant experience of working with crowded electronic interface in order to get the job done.</td>
</tr>
<tr>
<td>17</td>
<td>B</td>
<td>POST_BS_7</td>
<td>Teachers should formulate new instructional methods in the classroom to deal with increased workload, i.e. emails with students.</td>
</tr>
<tr>
<td>18</td>
<td>C</td>
<td>PRE_BS_1</td>
<td>TeleTOP is the mandated default learning environment in the university; all faculties and schools should make use of it.</td>
</tr>
<tr>
<td>19</td>
<td>C</td>
<td>PRE_BS_2</td>
<td>TeleTOP is already an existing resource for supporting educational tasks in the university and faculty. The school of industrial design engineering must make use of it.</td>
</tr>
<tr>
<td>20</td>
<td>C</td>
<td>EARLY_BS_1</td>
<td>There should be a TeleTOP expert available for school of Industrial Design Engineering to oversee the system deployment.</td>
</tr>
<tr>
<td>21</td>
<td>C</td>
<td>EARLY_BS_2</td>
<td>The TeleTOP expert should take pro-active efforts in generating support and stimulating software use in the school.</td>
</tr>
<tr>
<td>22</td>
<td>C</td>
<td>EARLY_BS_3</td>
<td>Stop using Workspace function for uploading projects and project files; Use the Workspace for submitting interim project files but not the final product.</td>
</tr>
<tr>
<td>23</td>
<td>C</td>
<td>EARLY_BS_4</td>
<td>Students should submit their assignments by hand. If agreed to be submitted online, it should be uploaded in TeleTOP at certain time period, i.e. 17.30</td>
</tr>
<tr>
<td>24</td>
<td>C</td>
<td>EARLY_BS_5</td>
<td>Students would like the teachers to give proper instructions on where to download class materials and lectures. If possible, use the different functions of TeleTOP that offer uploading and downloading possibilities should be consistent.</td>
</tr>
<tr>
<td>25</td>
<td>C</td>
<td>EARLY_BS_6</td>
<td>Make use of a student assistants to aide teachers in setting up TeleTOP sites for the course.</td>
</tr>
<tr>
<td>26</td>
<td>C</td>
<td>EARLY_BS_7</td>
<td>Retain the use of TeleTOP in the school despite its shortcomings.</td>
</tr>
<tr>
<td>27</td>
<td>C</td>
<td>POST_BS_1</td>
<td>Include the bureau of educational affairs in the re-design of TeleTOP and in managing the implementation.</td>
</tr>
<tr>
<td>28</td>
<td>C</td>
<td>POST_BS_2</td>
<td>There should be an official or formal introduction about the different software applications used and their abbreviations explained, i.e. TOST, TAST, VIST, TeleTOP.</td>
</tr>
<tr>
<td>29</td>
<td>C</td>
<td>POST_BS_3</td>
<td>Teachers maintain their own list and administration aside from the TeleTOP and VIST.</td>
</tr>
<tr>
<td>30</td>
<td>C</td>
<td>POST_BS_4</td>
<td>Utilize TeleTOP functions such as Poll in the classroom in order to determine students’ understanding of the learning material immediately.</td>
</tr>
<tr>
<td>31</td>
<td>D</td>
<td>PRE_BS_1</td>
<td>To address customer demands, FDS should set-up a training and documentation department: FDS Academy.</td>
</tr>
<tr>
<td>32</td>
<td>D</td>
<td>PRE_BS_2</td>
<td>Existing resources should be used i.e. in-house technical writer to lead the creation and set-up of the training department.</td>
</tr>
<tr>
<td>33</td>
<td>D</td>
<td>PRE_BS_3</td>
<td>Consultants with deep FDS system knowledge should become trainers and the vacant room in the building can become a training lab.</td>
</tr>
<tr>
<td>34</td>
<td>D</td>
<td>PRE_BS_4</td>
<td>Assistance of third party experts can be used to help develop the AC Academy training curriculum.</td>
</tr>
<tr>
<td>35</td>
<td>D</td>
<td>PRE_BS_5</td>
<td>Training will be class-room based, instructor-led training.</td>
</tr>
<tr>
<td>Item</td>
<td>Case</td>
<td>Code (R )</td>
<td>Requirements statement</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>36</td>
<td>D</td>
<td>PRE_BS_6</td>
<td>For the training lab, there should be one PC per trainee.</td>
</tr>
<tr>
<td>37</td>
<td>D</td>
<td>PRE_BS_7</td>
<td>Trainees will receive the training materials in electronic format on USB stick but excluding the FDS software.</td>
</tr>
<tr>
<td>38</td>
<td>D</td>
<td>PRE_BS_8</td>
<td>As business opportunities grow, FDS should create new functions and hire more staff.</td>
</tr>
<tr>
<td>39</td>
<td>D</td>
<td>EARLY_BS_1</td>
<td>FDS Academy would like to engage the services of a consulting company for the LMS selection process. A RFP was sent out with a list of functional requirements. See Table 8-3. Requirements derived from RFP.</td>
</tr>
<tr>
<td>40</td>
<td>D</td>
<td>EARLY_BS_2</td>
<td>FDS Academy should make use of trial account to create a demo environment that builds on existing training curriculum; This includes filling in template for site strategy, content, etc.</td>
</tr>
<tr>
<td>41</td>
<td>D</td>
<td>EARLY_BS_3</td>
<td>FDS Academy wants to sign Oracle iLearning hosting contract.</td>
</tr>
<tr>
<td>42</td>
<td>D</td>
<td>EARLY_BS_4</td>
<td>New employee training should also include a technical introduction to FDS products aside from company orientation. The technical training can be computer-based and conducted stand alone.</td>
</tr>
<tr>
<td>43</td>
<td>D</td>
<td>EARLY_BS_5</td>
<td>Classroom-based product training should also upgrade and innovate into blended learning.</td>
</tr>
<tr>
<td>44</td>
<td>D</td>
<td>EARLY_BS_6</td>
<td>Handouts and lecture slides should not be stored in the Oracle iLearning online server. They should be stored on a separate local server which interfaces with Oracle iLearning.</td>
</tr>
<tr>
<td>45</td>
<td>D</td>
<td>EARLY_BS_7</td>
<td>Updating existing slides and materials should be done via the local training server.</td>
</tr>
<tr>
<td>46</td>
<td>D</td>
<td>EARLY_BS_8</td>
<td>For onsite training, trainees will have access to the training slides through USB sticks that will be provided for them.</td>
</tr>
<tr>
<td>47</td>
<td>D</td>
<td>EARLY_BS_9</td>
<td>Assessment functionality in Oracle iLearning should be used to create course evaluation form.</td>
</tr>
<tr>
<td>48</td>
<td>D</td>
<td>EARLY_BS_10</td>
<td>Use MS Excel in processing raw course evaluation results, for distribution and archiving.</td>
</tr>
<tr>
<td>49</td>
<td>D</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
</tr>
<tr>
<td>50</td>
<td>D</td>
<td>EARLY_BS_12</td>
<td>The FDS Academy should be the system and site administrator of the FDS Oracle iLearning site. The training developer has content management rights.</td>
</tr>
<tr>
<td>51</td>
<td>D</td>
<td>EARLY_BS_13</td>
<td>The trainers do not have administrative rights, i.e. enrolment rights and content management rights. Trainer will have instructor rights (which mean no administrative rights, i.e. enrolment and content management).</td>
</tr>
<tr>
<td>52</td>
<td>D</td>
<td>EARLY_BS_14</td>
<td>The trainer should promote the use of Oracle iLearning in training sessions.</td>
</tr>
<tr>
<td>53</td>
<td>D</td>
<td>POST_BS_1</td>
<td>The new training coordinator should have administrative rights. Each training session should be set-up by the training coordinator in Oracle iLearning.</td>
</tr>
<tr>
<td>54</td>
<td>D</td>
<td>POST_BS_2</td>
<td>Customers and partners can make use of the e-learning modules in FDS as an interim solution for a training session that will come later.</td>
</tr>
<tr>
<td>55</td>
<td>D</td>
<td>POST_BS_3</td>
<td>Training coordinator would like to migrate course evaluation form after seeing the Marketing Department’s online survey environment.</td>
</tr>
<tr>
<td>56</td>
<td>D</td>
<td>POST_BS_4</td>
<td>Need for office space in home office prompts training center relocation to Amsterdam office.</td>
</tr>
<tr>
<td>57</td>
<td>D</td>
<td>POST_BS_5</td>
<td>FDS Academy will become FDS Training. FDS Training becomes part of beta-testing department, Solution Center. FDS Training is split into two locations: 2 team members remain in home office and 2 team members plus the manager work at the Amsterdam office.</td>
</tr>
<tr>
<td>58</td>
<td>D</td>
<td>POST_BS_6</td>
<td>The search for an open source LMS should be assigned to an intern. The intern can also make an inventory of LMS requirements for FDS Training. See Table 7-5 for the open LMS requirements inventory.</td>
</tr>
<tr>
<td>59</td>
<td>D</td>
<td>POST_BS_7</td>
<td>To cut down costs and remain viable, FDS has to reduce its staff. The training department has to be dissolved.</td>
</tr>
</tbody>
</table>
### Appendix B.3 Software Product Concept Domain Requirements Statements

<table>
<thead>
<tr>
<th>Item</th>
<th>Case</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>POST_SP_1</td>
<td>KENNISNET should be converted into a project management system.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>PRE_SP_1</td>
<td>The software has to be an e-learning software.</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>POST_SP_1</td>
<td>The system is the default communicator tool in the organization.</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>POST_SP_2</td>
<td>The software serves as a proxy for the virtual organization.</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>POST_SP_3</td>
<td>FCC supports e-learning and distance learning.</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>POST_SP_4</td>
<td>FCC serves as a ‘closed’ system exclusively accessible only to Of employees.</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>PRE_SP_1</td>
<td>TeleTOP is the tele-learning tool; it provides support in the teaching and learning process.</td>
</tr>
<tr>
<td>11</td>
<td>C</td>
<td>EARLY_SP_1</td>
<td>TeleTOP is an educational tool suitable for project-based education.</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>EARLY_SP_2</td>
<td>TeleTOP is an asymmetric and exclusive communication tool between teachers and students.</td>
</tr>
<tr>
<td>13</td>
<td>C</td>
<td>EARLY_SP_3</td>
<td>TeleTOP is a single course management and material repository system.</td>
</tr>
<tr>
<td>14</td>
<td>C</td>
<td>POST_SP_1</td>
<td>TeleTOP should evolve into an integrated learning suite incorporating multiple systems or their functionalities.</td>
</tr>
<tr>
<td>15</td>
<td>C</td>
<td>POST_SP_2</td>
<td>TeleTOP is an exclusive application for teachers and students only</td>
</tr>
<tr>
<td>16</td>
<td>C</td>
<td>POST_SP_3</td>
<td>TeleTOP should make itself more amenable to wireless capabilities and support mobile education.</td>
</tr>
<tr>
<td>17</td>
<td>C</td>
<td>POST_SP_4</td>
<td>In the light of the integration among the three technical universities, a new digital, integrated earning and content management system will replace TeleTOP.</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>EARLY_SP_1</td>
<td>FDS Academy, tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>EARLY_SP_2</td>
<td>Oracle iLearning proposed as affordable and suitable LMS alternative; FDS Academy should try out this product.</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>EARLY_SP_3</td>
<td>Oracle iLearning will be the default environment for delivering training in e-learning format; this is applicable especially to new employee training.</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning.</td>
</tr>
<tr>
<td>22</td>
<td>D</td>
<td>POST_SP_1</td>
<td>Oracle iLearning should be seen as a sales and account management support tool.</td>
</tr>
<tr>
<td>23</td>
<td>D</td>
<td>POST_SP_2</td>
<td>Course evaluation doesn’t have to take place in Oracle iLearning anymore.</td>
</tr>
<tr>
<td>24</td>
<td>D</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>25</td>
<td>D</td>
<td>POST_SP_4</td>
<td>With the dissolution of the training department, Oracle iLearning is no longer needed. Subscription has to be at minimum to meet contractual obligation with customers.</td>
</tr>
<tr>
<td>26</td>
<td>D</td>
<td>POST_SP_5</td>
<td>Other e-learning authoring tool, i.e. Adobe Captivate should be used for creating assessments instead of Oracle iLearning.</td>
</tr>
<tr>
<td>27</td>
<td>D</td>
<td>POST_SP_6</td>
<td>[OpenScrLMS] Course delivery: Training portal idea?</td>
</tr>
<tr>
<td>28</td>
<td>D</td>
<td>POST_SP_7</td>
<td>[OpenScrLMS] Technological: What to do with Dickens? meaning what is the implication to the existing training environment separate from Oracle iLearning.</td>
</tr>
</tbody>
</table>
### Appendix B.4 Software Solution Specification Domain Requirements Statements

<table>
<thead>
<tr>
<th>Item</th>
<th>Case</th>
<th>Code (R)</th>
<th>Requirements statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>PRE_SS_F_1</td>
<td>The system should store and remember data.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>PRE_SS_O_2</td>
<td>The system should be developed in Lotus Notes.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>EARLY_SS_F_1</td>
<td>The initial version of KENNISNET must be improved; it should have more functions and features.</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>EARLY_SS_F_2</td>
<td>The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop. See Figures 4-1 and 4-2.</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>EARLY_SS_F_3</td>
<td>Users should be informed of updates and changes.</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>EARLY_SS_F_4</td>
<td>The system should enable users to communicate directly with others, either one to one, or many to many.</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>EARLY_SS_F_5</td>
<td>KENNISNET should allow for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory (see Figure 4-2).</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>EARLY_SS_F_6</td>
<td>It allows users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>EARLY_SS_F_7</td>
<td>Users are able to know who have posted information, at what time and under which topic and theme.</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>EARLY_SS_F_8</td>
<td>The system should support two knowledge repositories: the knowledge bank and the experts directory. See Figure 4-2.</td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td>EARLY_SS_Q_9</td>
<td>There should be an online manual.</td>
</tr>
<tr>
<td>12</td>
<td>A</td>
<td>EARLY_SS_Q_10</td>
<td>The user should be able to trust the sources received.</td>
</tr>
<tr>
<td>13</td>
<td>A</td>
<td>EARLY_SS_Q_11</td>
<td>The system should be integrated with other knowledge systems such as ROLLS and DIAGNOSE.</td>
</tr>
<tr>
<td>14</td>
<td>A</td>
<td>EARLY_SS_Q_12</td>
<td>The system should run on Lotus Notes.</td>
</tr>
<tr>
<td>15</td>
<td>A</td>
<td>POST_SS_F_1</td>
<td>The system should provide notification for new entries added to the system.</td>
</tr>
<tr>
<td>16</td>
<td>A</td>
<td>POST_SS_F_2</td>
<td>The system should allow storage of WWW bookmarks.</td>
</tr>
<tr>
<td>17</td>
<td>A</td>
<td>POST_SS_F_3</td>
<td>The system should display or make prominent the latest information that is added on the system.</td>
</tr>
<tr>
<td>18</td>
<td>A</td>
<td>POST_SS_F_4</td>
<td>The system should provide links to information sources such as the Statistical Board.</td>
</tr>
<tr>
<td>19</td>
<td>A</td>
<td>POST_SS_Q_5</td>
<td>The system should be kept up-to-date.</td>
</tr>
<tr>
<td>20</td>
<td>A</td>
<td>POST_SS_Q_6</td>
<td>The interface design should be improved; it should be more pleasing aesthetically.</td>
</tr>
<tr>
<td>21</td>
<td>A</td>
<td>POST_SS_Q_7</td>
<td>Use a more advanced and intelligent search tool like Discovery where search and searcher profiles can be stored.</td>
</tr>
<tr>
<td>22</td>
<td>A</td>
<td>POST_SS_Q_8</td>
<td>The system should contain information obtained from the Association of Insurance Companies.</td>
</tr>
<tr>
<td>23</td>
<td>A</td>
<td>POST_SS_Q_9</td>
<td>The system should contain product information from each sub-company (brands, tariffs, premiums, claims, clients, internal figures) including planned and proposed.</td>
</tr>
<tr>
<td>24</td>
<td>A</td>
<td>POST_SS_O_10</td>
<td>Put the electronic handbooks or handbooks in the system.</td>
</tr>
<tr>
<td>25</td>
<td>B</td>
<td>PRE_SS_O_1</td>
<td>The system should run on Macintosh machines.</td>
</tr>
<tr>
<td>26</td>
<td>B</td>
<td>EARLY_SS_F_1</td>
<td>The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.</td>
</tr>
<tr>
<td>27</td>
<td>B</td>
<td>EARLY_SS_Q_1</td>
<td>FCC should also run on a Windows-based operating system.</td>
</tr>
<tr>
<td>28</td>
<td>B</td>
<td>EARLY_SS_Q_2</td>
<td>FCC has to be learnable for a new user within 14 days.</td>
</tr>
<tr>
<td>29</td>
<td>B</td>
<td>EARLY_SS_F_2</td>
<td>The software should facilitate online communication between teacher-trainers and students.</td>
</tr>
<tr>
<td>30</td>
<td>B</td>
<td>EARLY_SS_Q_3</td>
<td>The software has to be in the Dutch language.</td>
</tr>
<tr>
<td>31</td>
<td>B</td>
<td>EARLY_SS_Q_4</td>
<td>A Dutch version of a user manual should be made available.</td>
</tr>
<tr>
<td>32</td>
<td>B</td>
<td>POST_SS_Q_1</td>
<td>The system should have a means of filtering unnecessary information for the user or at least, it should direct users on how to filter information such that the proliferation of icons on the start-up screen of FCC is minimized.</td>
</tr>
<tr>
<td>33</td>
<td>C</td>
<td>PRE_SS_F_1</td>
<td>[ProdDesc] TeleTOP is a role-based web application; it recognizes two types of roles: teachers and students.</td>
</tr>
<tr>
<td>34</td>
<td>C</td>
<td>PRE_SS_F_2</td>
<td>[ProdDesc] Course information, registration and administration.</td>
</tr>
<tr>
<td>35</td>
<td>C</td>
<td>PRE_SS_F_3</td>
<td>[ProdDesc] Course planning, logistics and study support.</td>
</tr>
<tr>
<td>36</td>
<td>C</td>
<td>PRE_SS_F_4</td>
<td>[ProdDesc] Off classroom communication.</td>
</tr>
<tr>
<td>37</td>
<td>C</td>
<td>PRE_SS_F_5</td>
<td>[ProdDesc] Interactive online communication.</td>
</tr>
<tr>
<td>39</td>
<td>C</td>
<td>PRE_SS_Q_1</td>
<td>[ProdDesc] TeleTOP is available in both English and Dutch versions.</td>
</tr>
<tr>
<td>40</td>
<td>C</td>
<td>PRE_SS_Q_2</td>
<td>[ProdDesc] TeleTOP has a text-based manual available.</td>
</tr>
<tr>
<td>Item</td>
<td>Case</td>
<td>Code (R.)</td>
<td>Requirements statement</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>41</td>
<td>C</td>
<td>EARLY_SS_F_1</td>
<td>TeleTOP should adequately support uploading and storing of large files in the Workspace. There should be enough disk space to store the project files.</td>
</tr>
<tr>
<td>42</td>
<td>C</td>
<td>EARLY_SS_F_2</td>
<td>[ProdDesc] The workspace function of TeleTOP is improved. It can now allow submission of large files in the Workspace, i.e. 1 MB or more.</td>
</tr>
<tr>
<td>43</td>
<td>C</td>
<td>EARLY_SS_F_3</td>
<td>Functions that enable storage, uploading and downloading of files should be properly differentiated from each other. These functions are archive, presentations, roster and publications.</td>
</tr>
<tr>
<td>44</td>
<td>C</td>
<td>EARLY_SS_F_4</td>
<td>Users want a seamless integration of the files and objects stored in TeleTOP with the application</td>
</tr>
<tr>
<td>45</td>
<td>C</td>
<td>EARLY_SS_F_5</td>
<td>The files in the workspace should have a structure and hierarchy so that files can be searched systematically, i.e. per group, per topic, etc.</td>
</tr>
<tr>
<td>46</td>
<td>C</td>
<td>EARLY_SS_F_6</td>
<td>It should be possible to add the roster in a simpler way than the way it is: teachers have to fill in the schedule of the sessions line by line.</td>
</tr>
<tr>
<td>47</td>
<td>C</td>
<td>EARLY_SS_F_7</td>
<td>It should be possible to send an email to the whole group in one-click. (See Figure 7-5).</td>
</tr>
<tr>
<td>48</td>
<td>C</td>
<td>POST_SS_Q_1</td>
<td>There should only be one interface between TeleTOP, VIST, TOST, TAST and student portal.</td>
</tr>
<tr>
<td>49</td>
<td>C</td>
<td>POST_SS_F_1</td>
<td>Teachers should be able to create their own backup of Workspace files of their courses in DVD format.</td>
</tr>
<tr>
<td>50</td>
<td>C</td>
<td>POST_SS_F_2</td>
<td>It should be possible for courses in TeleTOP to be accessible on a read-only basis to unauthenticated users.</td>
</tr>
<tr>
<td>51</td>
<td>C</td>
<td>POST_SS_F_3</td>
<td>TeleTOP should provide support for mathematical symbols.</td>
</tr>
<tr>
<td>52</td>
<td>C</td>
<td>POST_SS_F_4</td>
<td>TeleTOP should display most recent items, i.e. news items, newly uploaded files</td>
</tr>
<tr>
<td>53</td>
<td>C</td>
<td>POST_SS_F_5</td>
<td>There should be online help available.</td>
</tr>
<tr>
<td>54</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>FDS should have a trial account to try out Oracle iLearning.</td>
</tr>
<tr>
<td>55</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Staff should receive training in Oracle iLearning.</td>
</tr>
<tr>
<td>56</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Oracle iLearning LMS site must conform to FDS look and feel</td>
</tr>
<tr>
<td>57</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Oracle iLearning should support multiple curricula, i.e. it should be possible to incorporate Powerpoint slides and exercises in pdf format used in classroom-based training.</td>
</tr>
<tr>
<td>58</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Oracle iLearning subscription is limited to 100 users. For each user, FDS is entitled to 5 MB of space.</td>
</tr>
<tr>
<td>59</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Hand-outs and lecture slides should be available in Oracle iLearning only for the duration of the training session. Trainee accounts should expire at the end of the training session.</td>
</tr>
<tr>
<td>60</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>It should be possible within Oracle iLearning to link to materials in external servers and display these correctly.</td>
</tr>
<tr>
<td>61</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Employee accounts do not expire.</td>
</tr>
<tr>
<td>62</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>The Oracle iLearning site for trainees at customer site will not contain powerpoint slides; however, they can view the training content outline.</td>
</tr>
<tr>
<td>63</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Course evaluation should be done via Oracle iLearning. Trainees should be able to fill-in an online course evaluation form.</td>
</tr>
<tr>
<td>64</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Reporting in Oracle iLearning requires knowledge of databases. Results of course evaluation are in raw format requiring transformation into other formats distributable per email.</td>
</tr>
<tr>
<td>65</td>
<td>D</td>
<td>EARLY_SS_F_1</td>
<td>Two new courses are added in the FDS curriculum. This should also be incorporated in the Oracle FDS iLearning site.</td>
</tr>
<tr>
<td>66</td>
<td>D</td>
<td>POST_SS_F_1</td>
<td>User subscription should be increased to 135 users.</td>
</tr>
<tr>
<td>67</td>
<td>D</td>
<td>POST_SS_Q_1</td>
<td>Training materials, including the FDS Oracle iLearning site should be consistent with the new housestyle.</td>
</tr>
<tr>
<td>68</td>
<td>D</td>
<td>POST_SS_F_2</td>
<td>Oracle iLearning subscription should be reduced to 100 users.</td>
</tr>
<tr>
<td>69</td>
<td>D</td>
<td>POST_SS_F_3</td>
<td>Employee accounts should expire. Accounts of those who have already undergone new employee training should be deleted.</td>
</tr>
<tr>
<td>70</td>
<td>D</td>
<td>EARLY_SS_Q_3</td>
<td>[RFP1] A hosted-solution, so that the company is not required to tax internal IT infrastructures to support the roll-out of the FDS Academy.</td>
</tr>
<tr>
<td>71</td>
<td>D</td>
<td>EARLY_SS_Q_4</td>
<td>[RFP2] Option of potentially relocating the solution to an internal web server in the future as demand increases.</td>
</tr>
<tr>
<td>72</td>
<td>D</td>
<td>EARLY_SS_F_12</td>
<td>[RFP3] Prefer an SQL database format (to be compliant with internal systems).</td>
</tr>
<tr>
<td>73</td>
<td>D</td>
<td>EARLY_SS_F_13</td>
<td>[RFP4] The company will give strong consideration to those vendors that have built-in authoring capability or can recommend a seamless content authoring approach. Asset Control is additionally considering the use of Trivantis, Lectora Publisher for content authoring and is still seeking a solution for rapid creation of software application simulations (i.e. RapidBuilder from XStream Software or OnDemand from Global Knowledge).</td>
</tr>
<tr>
<td>74</td>
<td>D</td>
<td>EARLY_SS_F_14</td>
<td>[RFP5] Performance tracking for asynchronous course material (using industry standards).</td>
</tr>
<tr>
<td>75</td>
<td>D</td>
<td>EARLY_SS_Q_5</td>
<td>[RFP6] A strong security module, allowing for appropriate access to course content and administrative data.</td>
</tr>
<tr>
<td>76</td>
<td>D</td>
<td>EARLY_SS_F_15</td>
<td>[RFP7] Threaded Discussion Groups (preferably linked to specific courses).</td>
</tr>
<tr>
<td>77</td>
<td>D</td>
<td>EARLY_SS_F_16</td>
<td>[RFP8] Ability to attach PowerPoint and Word Documents as reference material.</td>
</tr>
<tr>
<td>Item</td>
<td>Case</td>
<td>Code (R)</td>
<td>Requirements statement</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>78</td>
<td>D</td>
<td>EARLY_SS_F_17</td>
<td>[RFP9] Integrated email collaboration (learner to instructor, learner-to-learner, etc.).</td>
</tr>
<tr>
<td>80</td>
<td>D</td>
<td>EARLY_SS_Q_6</td>
<td>[RFP11] 24X7 support.</td>
</tr>
<tr>
<td>82</td>
<td>D</td>
<td>EARLY_SS_F_20</td>
<td>[RFP13] Level-1 Evaluation Utility (creating &quot;happy sheets&quot;).</td>
</tr>
<tr>
<td>83</td>
<td>D</td>
<td>EARLY_SS_Q_7</td>
<td>[RFP14] Regulatory Compliance.</td>
</tr>
<tr>
<td>84</td>
<td>D</td>
<td>EARLY_SS_F_21</td>
<td>[RFP15] Scheduling.</td>
</tr>
<tr>
<td>85</td>
<td>D</td>
<td>EARLY_SS_F_22</td>
<td>[RFP16] Doesn’t require a plug-in.</td>
</tr>
<tr>
<td>86</td>
<td>D</td>
<td>EARLY_SS_F_23</td>
<td>[RFP17] Classroom management (may use in the future, but not part of initial implementation).</td>
</tr>
<tr>
<td>88</td>
<td>D</td>
<td>EARLY_SS_F_25</td>
<td>[RFP19] Competency Management (future).</td>
</tr>
<tr>
<td>89</td>
<td>D</td>
<td>EARLY_SS_F_26</td>
<td>[RFP20] 360-degree evaluation (preferred for future).</td>
</tr>
<tr>
<td>91</td>
<td>D</td>
<td>EARLY_SS_Q_8</td>
<td>[RFP22] Even though FDS is located in the Netherlands, all training sessions will be conducted in English. There is no need for multi-lingual support or localization.</td>
</tr>
<tr>
<td>92</td>
<td>D</td>
<td>EARLY_SS_Q_9</td>
<td>[RFP23] The company has no immediate plans for providing e-commerce support (credit card transactions) in the initial implementation.</td>
</tr>
<tr>
<td>93</td>
<td>D</td>
<td>EARLY_SS_Q_10</td>
<td>[RFP24] There is no requirement at this time to integrate performance data with an external ERP or CRM system. This should be a fairly straightforward implementation of your hosted-LMS offering.</td>
</tr>
<tr>
<td>95</td>
<td>D</td>
<td>POST_SS_F_5</td>
<td>[OpenSrcLMS] Course delivery: PDF's usable (nice to have).</td>
</tr>
<tr>
<td>96</td>
<td>D</td>
<td>POST_SS_F_6</td>
<td>[OpenSrcLMS] Course delivery: Place to store assignments and reading material per lesson/module.</td>
</tr>
<tr>
<td>97</td>
<td>D</td>
<td>POST_SS_F_7</td>
<td>[OpenSrcLMS] Course delivery: Course calendar, preferably smoothly looking.</td>
</tr>
<tr>
<td>100</td>
<td>D</td>
<td>POST_SS_F_10</td>
<td>[OpenSrcLMS] Course delivery 4 Assessments: Types of test: With pictures?</td>
</tr>
<tr>
<td>104</td>
<td>D</td>
<td>POST_SS_F_14</td>
<td>[OpenSrcLMS] Course delivery 4 Assessments: gradebook per student.</td>
</tr>
<tr>
<td>105</td>
<td>D</td>
<td>POST_SS_F_15</td>
<td>[OpenSrcLMS] Course delivery 4 Assessments: Grading of coursework and roster processing, including waitlisting.</td>
</tr>
<tr>
<td>106</td>
<td>D</td>
<td>POST_SS_F_16</td>
<td>[OpenSrcLMS] Course delivery 4 Assessments: Assessments usable, preferably to import via SCORM /QTI.</td>
</tr>
<tr>
<td>107</td>
<td>D</td>
<td>POST_SS_F_17</td>
<td>[OpenSrcLMS] Course delivery 4 Assessments: Give feedback on tests.</td>
</tr>
<tr>
<td>108</td>
<td>D</td>
<td>POST_SS_F_18</td>
<td>[OpenSrcLMS] Course delivery 4 Assessments: Create reports on achievements of students on tests.</td>
</tr>
<tr>
<td>111</td>
<td>D</td>
<td>POST_SS_F_21</td>
<td>[OpenSrcLMS] Course delivery: Ability to create an index so that people can find a particular topic.</td>
</tr>
<tr>
<td>113</td>
<td>D</td>
<td>POST_SS_F_23</td>
<td>[OpenSrcLMS] Course delivery: Use of recorded video and audio?</td>
</tr>
<tr>
<td>114</td>
<td>D</td>
<td>POST_SS_F_24</td>
<td>[OpenSrcLMS] Course delivery: Training portal idea?</td>
</tr>
<tr>
<td>115</td>
<td>D</td>
<td>POST_SS_F_25</td>
<td>[OpenSrcLMS] Management: Manage users and groups.</td>
</tr>
<tr>
<td>116</td>
<td>D</td>
<td>POST_SS_F_26</td>
<td>[OpenSrcLMS] Management: Minimal 200 user accounts needed (over 130 in use now).</td>
</tr>
<tr>
<td>119</td>
<td>D</td>
<td>POST_SS_F_29</td>
<td>[OpenSrcLMS] Management: Use of grouping to allow students access to specific modules or curricula.</td>
</tr>
<tr>
<td>121</td>
<td>D</td>
<td>POST_SS_F_30</td>
<td>[OpenSrcLMS] Management: Be able to add modules/lessons to curricula.</td>
</tr>
<tr>
<td>Item</td>
<td>Case</td>
<td>Code (R)</td>
<td>Requirements Statement</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>122</td>
<td>D</td>
<td>POST_SS_F_31</td>
<td>[OpenSrcLMS] Management: Generate reports (on users (amount of logins per month) user's logon time, etc).</td>
</tr>
<tr>
<td>123</td>
<td>D</td>
<td>POST_SS_F_32</td>
<td>[OpenSrcLMS] Management: Multiple teachers per course – one single teachers account?</td>
</tr>
<tr>
<td>124</td>
<td>D</td>
<td>POST_SS_F_33</td>
<td>[OpenSrcLMS] Management: Auto enrollment, with coordinators confirmation (enrolling learners in courses when required according to predefined criteria, such as job title or work location).</td>
</tr>
<tr>
<td>125</td>
<td>D</td>
<td>POST_SS_F_34</td>
<td>[OpenSrcLMS] Communication / Interactivity: Learner messaging and notifications.</td>
</tr>
<tr>
<td>126</td>
<td>D</td>
<td>POST_SS_F_35</td>
<td>[OpenSrcLMS] Communication / Interactivity: Discussion forums?</td>
</tr>
<tr>
<td>128</td>
<td>D</td>
<td>POST_SS_F_36</td>
<td>[OpenSrcLMS] Technological: Accessible from outside the FDS-network.</td>
</tr>
<tr>
<td>130</td>
<td>D</td>
<td>POST_SS_Q_4</td>
<td>[OpenSrcLMS] Technological: No extra plugins needed, perhaps only flash or silverlight.</td>
</tr>
<tr>
<td>131</td>
<td>D</td>
<td>POST_SS_Q_5</td>
<td>[OpenSrcLMS] Technological: Security (to be defined by SA)?</td>
</tr>
<tr>
<td>133</td>
<td>D</td>
<td>POST_SS_Q_7</td>
<td>[OpenSrcLMS] Other: Solid structure of pages in the system.</td>
</tr>
<tr>
<td>134</td>
<td>D</td>
<td>POST_SS_C_1</td>
<td>[OpenSrcLMS] Other: Not more expensive than Oracle.</td>
</tr>
</tbody>
</table>
### M1. Business problem resolved by business solution

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Breakdown</td>
<td>PRE_BP_2</td>
<td>One of the goals of the organization is to build a community of employees and to evoke a feeling of ‘one AIG company’.</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_BP_3</td>
<td>The KCS - non-life insurance aims to translate the organizational goal of integrating and unifying the different sub-companies by developing a community of non-life insurance experts.</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>KCS wants to have a system that the non-life insurance experts will use.</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>KCS wants to have a system that the non-life insurance experts will use.</td>
<td>POST_BS_1</td>
<td>The users should be encouraged to make use of the system by organising meetings to talk about the system and finding ways to use it.</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
<td>POST_BS_1</td>
<td>The users should be encouraged to make use of the system by organising meetings to talk about the system and finding ways to use it.</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
<td>POST_BS_2</td>
<td>Users should put training and seminar materials in KENNISNET. If not available in electronic format, a brief summary of the training/seminars/courses should be written and be made available for other colleagues through KENNISNET.</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
<td>POST_BS_3</td>
<td>Users should publish in KENNISNET market research and competitor reports that were conducted in their sub-companies.</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
<td>POST_BS_4</td>
<td>Users should also publish work methods or processes of how a work problem or project was approached and solved.</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_BP_1</td>
<td>The institute wants to participate in the renewal and modernization efforts in the educational sector through ICT.</td>
<td>EARLY_BS_1</td>
<td>FCC will replace the currently existing Pegasus email system.</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_BP_3</td>
<td>IT Department and FCC administrators at OI. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.</td>
<td>EARLY_BS_1</td>
<td>FCC will replace the currently existing Pegasus email system.</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_BP_3</td>
<td>IT Department and FCC administrators at OI. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.</td>
<td>EARLY_BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
</tr>
<tr>
<td>12</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.</td>
<td>EARLY_BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
</tr>
<tr>
<td>13</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_BP_2</td>
<td>The teacher-trainers want to improve their competencies. Specifically, they want to improve</td>
<td>EARLY_BS_4</td>
<td>The IT department should also function as a helpdesk and provide technical assistance</td>
</tr>
</tbody>
</table>

1 For Case A, this is added or updated after the results of the second case, see Chapter 5, section 5.7.2
<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.</td>
<td>POST_BS_3</td>
<td>New employees should be trained on how to use FCC.</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_BP_2</td>
<td>The teacher-trainers want to improve their competencies. Specifically, they want to improve their knowledge and skills about FCC.</td>
<td>POST_BS_4</td>
<td>Apart from training, sessions should be given for advanced system functionalities and features, tips and tricks, as well as proper ways of using certain system functionalities.</td>
</tr>
<tr>
<td>16</td>
<td>B</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The teacher-trainers want to broaden their knowledge about the useful functions of the software in addition to the ones that they already know.</td>
<td>POST_BS_4</td>
<td>Apart from training, sessions should be given for advanced system functionalities and features, tips and tricks, as well as proper ways of using certain system functionalities.</td>
</tr>
<tr>
<td>17</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>Students should submit their assignments on time or ahead of time. The continued availability of TeleTOP brings the official notion of time. Students submit their assignments up to the last minute of the day; educational processes continue 24/7.</td>
<td>EARLY_BS_4</td>
<td>Students should submit their assignments by hand. If agreed to be submitted online, it should be uploaded in TeleTOP at certain time period, i.e. 17:30.</td>
</tr>
<tr>
<td>18</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_BP_3</td>
<td>The Roster is not filled up completely and consistently by teachers.</td>
<td>EARLY_BS_6</td>
<td>Make use of a student assistants to aide teachers in setting up TeleTOP sites for the course.</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_1</td>
<td>Aside from consulting services, customers demand for more structured and formal knowledge transfer efforts such as training and documentation regarding the use and implementation of FDS software.</td>
<td>PRE_BS_1</td>
<td>To address customer demands, FDS should set-up a training and documentation department; FDS Academy.</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_2</td>
<td>For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.</td>
<td>PRE_BS_2</td>
<td>Existing resources should be used i.e. in-house technical writer to lead the creation and set-up of the training department.</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_2</td>
<td>For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.</td>
<td>PRE_BS_3</td>
<td>Consultants with deep FDS system knowledge should become trainers and the vacant room in the building can become a training lab.</td>
</tr>
<tr>
<td>22</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_2</td>
<td>For the new training department, new competencies and resources are needed. These include trainers, training materials and a training lab.</td>
<td>PRE_BS_6</td>
<td>For the training lab, there should be one PC per trainer.</td>
</tr>
<tr>
<td>23</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_3</td>
<td>The new training department is not experienced with the provision of training services and development of training materials.</td>
<td>PRE_BS_4</td>
<td>Assistance of third party experts can be used to help develop the AC Academy training curriculum.</td>
</tr>
<tr>
<td>24</td>
<td>D</td>
<td>Initiative</td>
<td>PRE_BP_3</td>
<td>The new training department is not experienced with the provision of training services and development of training materials.</td>
<td>PRE_BS_5</td>
<td>Training will be class-room based, instructor-led training.</td>
</tr>
<tr>
<td>25</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_4</td>
<td>Training handouts should be available and accessible to trainees during and after the training.</td>
<td>PRE_BS_7</td>
<td>Trainees will receive the training materials in electronic format on USB stick but excluding the FDS software.</td>
</tr>
<tr>
<td>26</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_3</td>
<td>The new training department is not experienced with the provision of training services and development of training materials.</td>
<td>EARLY_BS_1</td>
<td>FDS Academy would like to engage the services of a consulting company for the LMS selection process. A RFP was sent out with a list of functional requirements. See Table 8-3, Requirements derived from RFP.</td>
</tr>
<tr>
<td>27</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_6</td>
<td>FDS feels the need to bring (new) employees up to speed in term of knowledge and competency.</td>
<td>EARLY_BS_4</td>
<td>New employee training should also include a technical introduction to FDS products aside from company orientation. The technical training can be computer-based and conducted stand alone.</td>
</tr>
<tr>
<td>28</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>Increasingly, Sales and US-based account managers continue to request for onsite training to customers who didn’t sign a software license agreement yet. Expressed knowledge gap by customers on FDS software is seen as a training need.</td>
<td>POST_BS_2</td>
<td>Customers and partners can make use of the e-learning modules in FDS as an interim solution for a training session that will come later.</td>
</tr>
<tr>
<td>Item (#)</td>
<td>Case</td>
<td>Trigger</td>
<td>Source (R1)</td>
<td>Requirements Statement</td>
<td>Update (R2)</td>
<td>Requirements Statement</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-----------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>29</td>
<td>D</td>
<td>Initiative</td>
<td>POST_BP_5</td>
<td>New FDS training manager wants to cut down costs.</td>
<td>POST_BS_6</td>
<td>The search for an open source LMS should be assigned to an intern. The intern can also make an inventory of LMS requirements for FDS Training. See Table 8-5 for the open LMS requirements inventory.</td>
</tr>
<tr>
<td>30</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_4</td>
<td>Quality of training deteriorates due to out of funds during onsite training. Trainer reports the course evaluation takes too long to load and often leads to time outs.</td>
<td>POST_BS_3</td>
<td>Training coordinator would like to migrate course evaluation form after seeing the Marketing Department's online survey environment.</td>
</tr>
<tr>
<td>31</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_10</td>
<td>FDS is not able to sign any license agreement in the last 6 months of 2008. The prognosis for next year is zero sales. Customers are canceling maintenance and support agreements.</td>
<td>POST_BS_7</td>
<td>To cut down costs and remain viable, FDS has to reduce its staff. The training department has to be dissolved.</td>
</tr>
</tbody>
</table>
**M2. Business solution motivating a software product concept**

<table>
<thead>
<tr>
<th>Item (n)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building.</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>POST_SP_1</td>
<td>KENNISNET should be converted into a project management system.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>FCC will replace the currently existing Pegasus email system.</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>Initiative</td>
<td>EARLY_BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
<td>POST_SP_2</td>
<td>The software serves as a proxy for the virtual organization.</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>Initiative</td>
<td>EARLY_BS_3</td>
<td>All new teacher-trainers at OI have 14 days to familiarize themselves with FCC.</td>
<td>POST_SP_2</td>
<td>The software serves as a proxy for the virtual organization.</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>Initiative</td>
<td>EARLY_BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
<td>POST_SP_4</td>
<td>FCC serves as a ‘closed’ system exclusively accessible only to OI employees.</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>Initiative</td>
<td>POST_BS_1</td>
<td>All teacher-trainers should begin using FCC should in communicating with their students.</td>
<td>POST_SP_3</td>
<td>FCC supports e-learning and distance learning.</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>Initiative</td>
<td>POST_BS_2</td>
<td>The delivery of instructional materials to students should be in electronic form and disseminated through FCC.</td>
<td>POST_SP_3</td>
<td>FCC supports e-learning and distance learning.</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>Initiative</td>
<td>POST_BS_3</td>
<td>New employees should be trained on how to use FCC.</td>
<td>POST_SP_2</td>
<td>The software serves as a proxy for the virtual organization.</td>
</tr>
<tr>
<td>11</td>
<td>C</td>
<td>Initiative</td>
<td>EARLY_BS_2</td>
<td>The TelTOP expert should take pro-active efforts in generating support and stimulating software use in the school.</td>
<td>EARLY_SP_1</td>
<td>TelTOP is an educational tool suitable for project-based education.</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Initiative</td>
<td>EARLY_BS_7</td>
<td>Retain the use of TelTOP in the school despite its shortcomings.</td>
<td>POST_SP_3</td>
<td>TelTOP should make itself more amenable to wireless capabilities and support mobile education.</td>
</tr>
<tr>
<td>13</td>
<td>C</td>
<td>Initiative</td>
<td>EARLY_BS_7</td>
<td>Retain the use of TelTOP in the school despite its shortcomings.</td>
<td>POST_SP_4</td>
<td>In the light of the integration among the three technical universities, a new digital, integrated learning and content management system will replace TelTOP.</td>
</tr>
<tr>
<td>14</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BS_5</td>
<td>Training will be class-room based, instructor-led training.</td>
<td>EARLY_SP_1</td>
<td>FDS Academy tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
</tr>
</tbody>
</table>

2 For Case A, this is added or updated after the results of the second case, see Chapter 5, section 5.7.2
<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BS_5</td>
<td>Training will be class-room based, instructor-led training.</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning.</td>
</tr>
<tr>
<td>16</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_2</td>
<td>FDS Academy should make use of trial account to create a demo environment that builds on existing training curriculum. This includes filling in template for site strategy, content, etc.</td>
<td>EARLY_SP_3</td>
<td>Oracle iLearning will be the default environment for delivering training in e-learning format; this is applicable especially to new employee training.</td>
</tr>
<tr>
<td>17</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_4</td>
<td>New employee training should also include a technical introduction to FDS products aside from company orientation. The technical training can be computer-based and conducted stand alone.</td>
<td>EARLY_SP_3</td>
<td>Oracle iLearning will be the default environment for delivering training in e-learning format; this is applicable especially to new employee training.</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_5</td>
<td>Classroom-based product training should also upgrade and innovate into blended learning.</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning.</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>Initiative</td>
<td>POST_BS_2</td>
<td>Customers and partners can make use of the e-learning modules in FDS as an interim solution for a training session that will come later.</td>
<td>POST_SP_1</td>
<td>Oracle iLearning should be seen as a sales and account management support tool.</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>Initiative</td>
<td>POST_BS_3</td>
<td>Training coordinator would like to migrate course evaluation form after seeing the Marketing Department’s online survey environment.</td>
<td>POST_SP_2</td>
<td>Course evaluation doesn’t have to take place in Oracle iLearning anymore.</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BS_7</td>
<td>To cut down costs and remain viable, FDS has to reduce its staff. The training department has to be dissolved.</td>
<td>POST_SP_4</td>
<td>With the dissolution of the training department, Oracle iLearning is no longer needed. Subscription has to be at minimum to meet contractual obligation with customers.</td>
</tr>
</tbody>
</table>
### M3. Software product concept realized by a software solution specification

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger¹</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>PRE_SS_F_1</td>
<td>The system should store and remember data.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>PRE_SS_O_2</td>
<td>The system should be developed in Lotus Notes.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_F_2</td>
<td>The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_F_3</td>
<td>Users should be informed of updates and changes.</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_F_4</td>
<td>The system should enable users to communicate directly with others, either one to one, or many to many.</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_F_5</td>
<td>KENNISNET should allow for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory.</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_F_6</td>
<td>It should allow users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_F_7</td>
<td>Users are able to know who have posted information, at what time and under which topic and theme.</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_F_8</td>
<td>The system should support two knowledge repositories: the knowledge bank and the experts directory.</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_Q_9</td>
<td>There should be an online manual.</td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_Q_10</td>
<td>The user should be able to trust the sources received.</td>
</tr>
</tbody>
</table>

³ For Case A, this is added or updated after the results of the second case, see Chapter 5, section 5.7.2
<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_O_11</td>
<td>The system should be integrated with other knowledge systems such as ROLLS and DIAGNOSE.</td>
</tr>
<tr>
<td>13</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The system should be a knowledge management system for enabling knowledge exchange.</td>
<td>EARLY_SS_O_12</td>
<td>The system should run on Lotus Notes.</td>
</tr>
<tr>
<td>14</td>
<td>A</td>
<td>Initiative</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
<td>POST_SS_O_7</td>
<td>Use a more advanced and intelligent search tool like Discovery where search and searcher profiles can be stored.</td>
</tr>
<tr>
<td>15</td>
<td>A</td>
<td>Initiative</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
<td>POST_SS_O_8</td>
<td>The system should contain information obtained from the Association of Insurance Companies.</td>
</tr>
<tr>
<td>16</td>
<td>A</td>
<td>Initiative</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
<td>POST_SS_O_9</td>
<td>The system should contain product information from each sub-company (brands, tariffs, premiums, claims, clients, internal figures) including planned and proposed.</td>
</tr>
<tr>
<td>17</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>The software has to be an e-learning software.</td>
<td>EARLY_SS_F_2</td>
<td>The software should facilitate online communication between teacher-trainers and students.</td>
</tr>
<tr>
<td>18</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
<td>EARLY_SS_F_1</td>
<td>The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.</td>
</tr>
<tr>
<td>19</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
<td>EARLY_SS_F_2</td>
<td>The software should facilitate online communication between teacher-trainers and students.</td>
</tr>
<tr>
<td>20</td>
<td>C</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>TeleTOP is the tele-learning tool: it provides support in the teaching and learning process.</td>
<td>PRE_SS_[ProdDesc]</td>
<td>Product description of TeleTOP at pre-implementation listed in Tables 7-3 &amp; 7-4.</td>
</tr>
<tr>
<td>21</td>
<td>C</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>TeleTOP is the tele-learning tool: it provides support in the teaching and learning process.</td>
<td>EARLY_SS_F_1</td>
<td>TeleTOP should adequately support uploading and storing of large files in the Workspace. There should be enough disk space to store the project files.</td>
</tr>
<tr>
<td>22</td>
<td>C</td>
<td>Initiative</td>
<td>PRE_SP_1</td>
<td>TeleTOP is the tele-learning tool: it provides support in the teaching and learning process.</td>
<td>EARLY_SS_F_7</td>
<td>It should be possible to send an email to the whole group in one-click. (see Figure 7-5).</td>
</tr>
<tr>
<td>23</td>
<td>C</td>
<td>Initiative</td>
<td>EARLY_SP_1</td>
<td>TeleTOP is an educational tool suitable for project-based education.</td>
<td>EARLY_SS_F_1</td>
<td>TeleTOP should adequately support uploading and storing of large files in the Workspace. There should be enough disk space to store the project files.</td>
</tr>
<tr>
<td>24</td>
<td>C</td>
<td>Initiative</td>
<td>EARLY_SP_1</td>
<td>TeleTOP is an educational tool suitable for project-based education.</td>
<td>EARLY_SS_F_2</td>
<td>[ProdDesc] The workspace function of TeleTOP is improved. It can now allow submission of large files in the Workspace, i.e. 1 MB or more.</td>
</tr>
<tr>
<td>25</td>
<td>C</td>
<td>Initiative</td>
<td>EARLY_SP_2</td>
<td>TeleTOP is an asymmetric and exclusive communication tool between teachers and students.</td>
<td>EARLY_SS_F_7</td>
<td>It should be possible to send an email to the whole group in one-click. (see Figure 7-5).</td>
</tr>
<tr>
<td>26</td>
<td>C</td>
<td>Initiative</td>
<td>POST_SP_1</td>
<td>TeleTOP should evolve into an integrated learning suite incorporating multiple systems or their functionalities.</td>
<td>POST_SS_Q_1</td>
<td>There should only be one interface between TeleTOP, VIST, TOST, TAST and student portal.</td>
</tr>
<tr>
<td>27</td>
<td>C</td>
<td>Initiative</td>
<td>POST_SP_2</td>
<td>TeleTOP is an exclusive application for teachers and students only</td>
<td>POST_SS_F_2</td>
<td>It should be possible for courses in TeleTOP to be accessible on a read-only basis to unauthenticated users.</td>
</tr>
<tr>
<td>28</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SP_1</td>
<td>FDS Academy: tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
<td>EARLY_SS_Q_3</td>
<td>[RFP1] A hosted-solution, so that the company is not required to tax internal IT infrastructures to support the roll-out of the FDS Academy.</td>
</tr>
<tr>
<td>Item (#)</td>
<td>Case</td>
<td>Trigger</td>
<td>Source (R1)</td>
<td>Requirements Statement</td>
<td>Update (R2)</td>
<td>Requirements Statement</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>---------</td>
<td>-------------</td>
<td>-----------------------</td>
<td>-------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>29</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SP_1</td>
<td>FDS Academy, tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
<td>EARLY_SS_Q_4</td>
<td>RFP2] Option of potentially relocating the solution to an internal web server in the future as demand increases.</td>
</tr>
<tr>
<td>30</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SP_1</td>
<td>FDS Academy, tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
<td>EARLY_SS_F_7</td>
<td>Employee accounts do not expire.</td>
</tr>
<tr>
<td>31</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SP_2</td>
<td>Oracle iLearning proposed as affordable and suitable LMS alternative; FDS Academy should try out this product.</td>
<td>EARLY_SS_F_1</td>
<td>FDS should have a trial account to try out Oracle iLearning.</td>
</tr>
<tr>
<td>32</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SP_2</td>
<td>Oracle iLearning proposed as affordable and suitable LMS alternative; FDS Academy should try out this product.</td>
<td>EARLY_SS_Q_1</td>
<td>Staff should receive training in Oracle Learning.</td>
</tr>
<tr>
<td>33</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SP_3</td>
<td>Oracle iLearning will be the default environment for delivering training in e-learning format; this is applicable especially to new employee training.</td>
<td>EARLY_SS_Q_2</td>
<td>Oracle iLearning LMS site must conform to FDS look and feel</td>
</tr>
<tr>
<td>34</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning.</td>
<td>EARLY_SS_F_3</td>
<td>Oracle iLearning should support multiple curricula, i.e. it should be possible to incorporate Powerpoint slides and exercises in pdf format used in classroom-based training.</td>
</tr>
<tr>
<td>35</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning.</td>
<td>EARLY_SS_F_5</td>
<td>Hand-outs and lecture slides should be available in Oracle iLearning only for the duration of the training session. Trainee accounts should expire at the end of the training session.</td>
</tr>
<tr>
<td>36</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SP_4</td>
<td>As e-learning software, Oracle iLearning LMS will support blended learning.</td>
<td>EARLY_SS_F_13</td>
<td>[RFP4] The company will give strong consideration to those vendors that have built-in authoring capability or can recommend a seamless content authoring approach. Asset Control is additionally considering the use of Thrantis, Lectora Publisher for content authoring and is still seeking a solution for rapid creation of software application simulations (i.e. RapidBuilder from XStream Software or OnDemand from Global Knowledge).</td>
</tr>
<tr>
<td>37</td>
<td>D</td>
<td>Initiative</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS</td>
<td>POST_SS_F_3</td>
<td>Employee accounts should expire. Accounts of those who have already undergone new employee training should be deleted.</td>
</tr>
<tr>
<td>38</td>
<td>D</td>
<td>Initiative</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS</td>
<td>POST_[OpenSrcLMS]*</td>
<td>All statements in the open source LMS project RFP listed in post deployment.</td>
</tr>
<tr>
<td>39</td>
<td>D</td>
<td>Initiative</td>
<td>POST_SP_5</td>
<td>Other e-learning authoring tool, i.e. Adobe Captivate should be used for creating assessments instead of Oracle iLearning.</td>
<td>POST_SS_F_13</td>
<td>[OpenSrcLMS] Course delivery ▶ Assessments: Import test from Oracle/Captivate</td>
</tr>
</tbody>
</table>
### M4. Software solution specification supported by business solutions

<table>
<thead>
<tr>
<th>Item</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_1</td>
<td>The initial version of KEENISNET must be improved; it should have more functions and features.</td>
<td>EARLY_SS_F_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SS_F_2</td>
<td>The software should facilitate online communication between teacher-trainers and students.</td>
<td>EARLY_SS_F_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SS_F_2</td>
<td>The software should facilitate online communication between teacher-trainers and students.</td>
<td>POST_SS_2</td>
<td>The delivery of instructional materials to students should be in electronic form and disseminated through FCC.</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SS_Q_3</td>
<td>The software has to be in the Dutch language.</td>
<td>POST_SS_3</td>
<td>New employees should be trained on how to use FCC.</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>Breakdown</td>
<td>POST_SS_Q_1</td>
<td>There should be a way to minimize the proliferation of icons on the start-up screen of FCC.</td>
<td>POST_SS_5</td>
<td>Users should organize and sort information on their screen in order to avoid information clutter.</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>Breakdown</td>
<td>POST_SS_Q_1</td>
<td>There should be a way to minimize the proliferation of icons on the start-up screen of FCC.</td>
<td>POST_SS_6</td>
<td>Do nothing: continue using FCC despite the unpleasant experience of working with crowded electronic interface in order to get the job done.</td>
</tr>
<tr>
<td>7</td>
<td>C</td>
<td>Initiative</td>
<td>PRE_SS_[ProdDesc]</td>
<td>Product description of TeleTOP at pre-implementation listed in Tables 7-3 &amp; 7-4.</td>
<td>PRE_SS_2</td>
<td>TeleTOP is already an existing resource for supporting educational tasks in the university and faculty. The school of industrial design engineering must make use of it.</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_SS_F_1</td>
<td>TeleTOP should adequately support uploading and storing of large files in the Workspace. There should be enough disk space to store the project files.</td>
<td>EARLY_SS_3</td>
<td>Stop using Workspace function for uploading projects and project files; Use the Workspace for submitting interim project files but not the final product.</td>
</tr>
<tr>
<td>9</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_SS_F_3</td>
<td>Functions that enable storage, uploading and downloading of files should be properly differentiated from each other. These functions are archive, presentations, roster and publications.</td>
<td>EARLY_SS_5</td>
<td>Students would like the teachers to give proper instructions on where to download class materials and lectures. If possible, the use of the different functions of TeleTOP that offer uploading and downloading possibilities should be consistent.</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_SS_F_6</td>
<td>It should be possible to fill-up the roster in a simpler way than the way it is: teachers have to fill in the schedule of the sessions line by line.</td>
<td>EARLY_SS_6</td>
<td>Make use of a student assistants to aide teachers in setting up TeleTOP sites for the course.</td>
</tr>
<tr>
<td>11</td>
<td>C</td>
<td>Initiative</td>
<td>PRE_SS_F_6</td>
<td>[ProdDesc] Collaboration</td>
<td>POST_SS_4</td>
<td>Utilize TeleTOP functions such as Poll in the classroom in order to determine students’ understanding of the learning material immediately.</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_1</td>
<td>FDS should have a trial account to try out Oracle iLearning.</td>
<td>EARLY_SS_2</td>
<td>FDS Academy should make use of trial account to create a demo environment that builds on existing training curriculum; This includes filling in template for site strategy, content, etc.</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_1</td>
<td>FDS should have a trial account to try out Oracle iLearning.</td>
<td>EARLY_SS_3</td>
<td>FDS Academy wants to sign Oracle iLearning hosting contract.</td>
</tr>
<tr>
<td>14</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_[RFP]*</td>
<td>All statements in the RFP listed in early implementation.</td>
<td>EARLY_SS_3</td>
<td>FDS Academy wants to sign Oracle iLearning hosting contract.</td>
</tr>
</tbody>
</table>

---

4 This is added or updated after the results of the second case, see Chapter 5, section 5.7.2
<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_4</td>
<td>Oracle iLearning subscription is limited to 100 users. For each user, FDS is entitled to 5 MB of space.</td>
<td>EARLY_BS_6</td>
<td>Handouts and lecture slides should not be stored in the Oracle iLearning online server. They should be stored on a separate local server which interfaces with Oracle iLearning.</td>
</tr>
<tr>
<td>16</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_4</td>
<td>Oracle iLearning subscription is limited to 100 users. For each user, FDS is entitled to 5 MB of space.</td>
<td>EARLY_BS_7</td>
<td>Updating existing slides and materials should be done via the local training server.</td>
</tr>
<tr>
<td>17</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_8</td>
<td>The Oracle iLearning site for trainees at customer site will not contain powerpoint slides; however, they can view the training content outline.</td>
<td>EARLY_BS_8</td>
<td>For onsite training, trainees will have access to the training slides through USB sticks that will be provided for them.</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SS_F_9</td>
<td>Course evaluation should be done via Oracle iLearning. Trainees should be able to fill in an online course evaluation form.</td>
<td>EARLY_BS_9</td>
<td>Assessment functionality in Oracle iLearning should be used to create course evaluation form.</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_10</td>
<td>Reporting in Oracle iLearning requires knowledge of databases. Results of course evaluation are in raw format requiring transformation into other formats distributable per email.</td>
<td>EARLY_BS_10</td>
<td>Use MS Excel in processing raw course evaluation results, for distribution and archiving.</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_9</td>
<td>Course evaluation should be done via Oracle iLearning. Trainees should be able to fill in an online course evaluation form.</td>
<td>POST_BS_3</td>
<td>Training coordinator would like to migrate course evaluation form after seeing the Marketing Department's online survey environment.</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SS_F_13</td>
<td>[RFP4] The company will give strong consideration to those vendors that have built-in authoring capability or can recommend a seamless content authoring approach. Asset Control is additionally considering the use of Tivantis, Lectora Publisher for content authoring and is still seeking a solution for rapid creation of software application simulations (i.e. RapidBuilder from Xstream Software or OnDemand from Global Knowledge).</td>
<td>POST_BS_2</td>
<td>Customers and partners can make use of the e-learning modules in FDS as an interim solution for a training session that will come later.</td>
</tr>
</tbody>
</table>
M5. Business solutions leading to improved software solution specifications

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building.</td>
<td>EARLY_SS_F_2</td>
<td>The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Initiative</td>
<td>PRE_BS_1</td>
<td>The adoption of a Knowledge Management (KM) strategy is a means of community-building.</td>
<td>EARLY_SS_F_4</td>
<td>The system should enable users to communicate directly with others, either one to one, or many to many.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_F_2</td>
<td>The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_F_3</td>
<td>Users should be informed of updates and changes.</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_F_4</td>
<td>The system should enable users to communicate directly with others, either one to one, or many to many.</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_F_5</td>
<td>KENNSINET should allow for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory.</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_F_6</td>
<td>It allows users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_F_7</td>
<td>Users are able to know who have posted information, at what time and under which topic and theme; EARLY_SS_F_8: The system should support two knowledge repositories: the knowledge bank and the experts directory.</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_F_8</td>
<td>The system should support two knowledge repositories: the knowledge bank and the experts directory.</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_Q_9</td>
<td>There should be an online manual.</td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_Q_10</td>
<td>The user should be able to trust the sources received.</td>
</tr>
</tbody>
</table>

5 For Case A, this is added or updated after the results of the second case, see Chapter 5, section 5.7.2
<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_Q_11</td>
<td>The system should be integrated with other knowledge systems such as ROLLs and DIAGNOSE.</td>
</tr>
<tr>
<td>13</td>
<td>A</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>The end-users should be involved in the re-design process in order to come up with more definite requirements.</td>
<td>EARLY_SS_Q_12</td>
<td>The system should run on Lotus Notes.</td>
</tr>
<tr>
<td>14</td>
<td>B</td>
<td>Initiative</td>
<td>EARLY_BS_2</td>
<td>Communication and exchange of materials among teacher-trainers should take place through FCC.</td>
<td>EARLY_SS_F_1</td>
<td>The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>Initiative</td>
<td>EARLY_BS_3</td>
<td>All new teacher-trainers at O have 14 days to familiarize themselves with FCC.</td>
<td>EARLY_SS_Q_2</td>
<td>FCC has to be learnable for a new user within 14 days.</td>
</tr>
<tr>
<td>16</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_BS_3</td>
<td>Students would like the teachers to give proper instructions on where to download class materials and lectures. If possible, the use of the different functions of TeleTOP that offer uploading and downloading possibilities should be consistent.</td>
<td>EARLY_SS_F_2</td>
<td>[ProdDesc] The workspace function of TeleTOP is improved. It can now allow submission of large files in the Workspace, i.e. 1 MB or more.</td>
</tr>
<tr>
<td>17</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_BS_5</td>
<td>All statements in the RFP listed in early implementation.</td>
<td>POST_SS_Q_2</td>
<td>There should be online help available.</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_1</td>
<td>FDS Academy would like to engage the services of a consulting company for the LMS selection process. A RFP was sent out with a list of functional requirements. See Table 8-3. Requirements derived from RFP.</td>
<td>EARLY_[RFP]</td>
<td>Staff should receive training in Oracle iLearning.</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_3</td>
<td>FDS Academy wants to sign Oracle iLearning hosting contract.</td>
<td>EARLY_SS_Q_1</td>
<td>Oracle iLearning LMS site must conform to FDS look and feel.</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_3</td>
<td>FDS Academy wants to sign Oracle iLearning hosting contract.</td>
<td>EARLY_SS_Q_2</td>
<td>Oracle iLearning LMS site must conform to FDS look and feel.</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_4</td>
<td>New employee training should also include a technical introduction to FDS products aside from company orientation. The technical training can be computer-based and conducted stand-alone.</td>
<td>EARLY_SS_F_13</td>
<td>[RFP#] The company will give strong consideration to those vendors that have built-in authoring capability or can recommend a seamless content authoring approach. Asset Control is additionally considering the use of Trivantis, Lectora Publisher for content authoring and is still seeking a solution for rapid creation of software application simulations (i.e. RapidBuilder from XStream Software or OnDemand from Global Knowledge).</td>
</tr>
<tr>
<td>22</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_4</td>
<td>New employee training should also include a technical introduction to FDS products aside from company orientation. The technical training can be computer-based and conducted stand-alone.</td>
<td>EARLY_SS_F_7</td>
<td>Employee accounts do not expire.</td>
</tr>
<tr>
<td>23</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_5</td>
<td>Classroom-based product training should also upgrade and innovate into blended learning.</td>
<td>EARLY_SS_F_3</td>
<td>Oracle iLearning should support multiple curricula, i.e. it should be possible to incorporate Powerpoint slides and exercises in pdf format used in classroom-based training.</td>
</tr>
<tr>
<td>24</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_5</td>
<td>Classroom-based product training should also upgrade and innovate into blended learning.</td>
<td>EARLY_SS_F_16</td>
<td>[RFP#] Ability to attach PowerPoint and Word Documents as reference material.</td>
</tr>
<tr>
<td>25</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_5</td>
<td>Classroom-based product training should also upgrade and innovate into blended learning.</td>
<td>EARLY_SS_F_5</td>
<td>Hand-outs and lecture slides should be available in Oracle iLearning only for the duration of the training session. Trainee accounts should expire at the end of the training session.</td>
</tr>
<tr>
<td>26</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_BS_6</td>
<td>Handouts and lecture slides should be stored in the Oracle iLearning online server. They should be stored on a separate local server which interfaces with Oracle Learning.</td>
<td>EARLY_SS_F_6</td>
<td>It should be possible within Oracle iLearning to link to materials in external servers and display these correctly.</td>
</tr>
<tr>
<td>27</td>
<td>D</td>
<td>Initiative</td>
<td>POST_BS_6</td>
<td>The search for an open source LMS should be assigned to an intern. The intern can</td>
<td>POST_SS_Open</td>
<td>All statements in the open source LMS project listed in post deployment (Table 8-4).</td>
</tr>
<tr>
<td>Item #</td>
<td>Case</td>
<td>Trigger</td>
<td>Source (R1)</td>
<td>Requirements Statement</td>
<td>Update (R2)</td>
<td>Requirements Statement</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>---------</td>
<td>-------------</td>
<td>------------------------</td>
<td>-------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>28</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>EARLY_SS_F_19</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Types of test: True / False</td>
</tr>
<tr>
<td>29</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_8</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Types of test: True / False</td>
</tr>
<tr>
<td>30</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_9</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Types of test: Multiple choice</td>
</tr>
<tr>
<td>31</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_10</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Types of test: With pictures?</td>
</tr>
<tr>
<td>32</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_11</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Types of test: Fill in the blanks</td>
</tr>
<tr>
<td>33</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_12</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Display scores and transcripts.</td>
</tr>
<tr>
<td>34</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_13</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Import test from Oracle/Captivate</td>
</tr>
<tr>
<td>35</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_14</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: gradebook per student</td>
</tr>
<tr>
<td>36</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_15</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Grading of coursework and roster processing, including waitlisting</td>
</tr>
<tr>
<td>37</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_16</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Assessments usable, preferably to import via SCORM QTI</td>
</tr>
<tr>
<td>38</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_17</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Give feedback on tests</td>
</tr>
<tr>
<td>39</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BS_11</td>
<td>Assessments should be integrated in the blended learning program. Each module preferably should conclude with an assessment.</td>
<td>POST_SS_F_18</td>
<td>[OpenSrcLMS] Course delivery ★ Assessments: Create reports on achievements of students on tests</td>
</tr>
</tbody>
</table>
### M6. Software solution specifications leading to new business problems

<table>
<thead>
<tr>
<th>Item</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_2</td>
<td>The system should facilitate knowledge exchange by enabling users to upload information items (data, text), to describe and classify these using the knowledge taxonomy that was developed during the design workshop.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_3</td>
<td>Users should be informed of updates and changes.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_4</td>
<td>The system should enable users to communicate directly with others, either one to one, or many to many.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_5</td>
<td>KENNISNET should allow for information search on (i) specific information search through the Knowledge Bank and (ii) search for experts in the Experts Directory.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_6</td>
<td>It allows users to electronically publish and upload information in the form of news, questions, discussion and documents; users can also post responses on these items.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_7</td>
<td>Users are able to know who have posted information, at what time and under which topic and theme.; EARLY_SS_F_8: &quot;The system should support two knowledge repositories: the knowledge bank and the experts directory.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_F_8</td>
<td>The system should support two knowledge repositories: the knowledge bank and the experts directory.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_Q_9</td>
<td>There should be an online manual.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_Q_10</td>
<td>The user should be able to trust the sources received.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_Q_11</td>
<td>The system should be integrated with other knowledge systems such as ROLLS and DIAGNOSE.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td>Breakdown</td>
<td>EARLY_SS_Q_12</td>
<td>The system should run on Lotus Notes.</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
</tr>
<tr>
<td>12</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SS_F_1</td>
<td>The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.</td>
<td>POST_BP_4</td>
<td>Users get the feeling that work never stops and that there is always a sense of urgency to reply to queries.</td>
</tr>
<tr>
<td>13</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SS_Q_2</td>
<td>FCC has to be learnable for a new user within 14 days.</td>
<td>POST_BP_1</td>
<td>The teacher-trainers want to broaden their about the useful functions of the software in addition to the ones that they already know.</td>
</tr>
</tbody>
</table>

---

6 For Case A, this is added or updated after the results of the second case, see Chapter 5, section 5.7.2
<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (Rt)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_SS_F_1</td>
<td>The software should facilitate a convenient 24/7 and economical means of communication between the teacher-trainers and institute, and between teacher-trainers themselves.</td>
<td>POSTBP_5</td>
<td>Users do not want to be flooded with unnecessary information on their FCC start-up screen.</td>
</tr>
<tr>
<td>15</td>
<td>C</td>
<td>Breakdown</td>
<td>PRE_SS_F_3</td>
<td>[ProdDesc] Course planning, logistics and study support</td>
<td>EARLY_BP_1</td>
<td>Students should submit their assignments on time or ahead of time. The continued availability of TeleTOP blurs the official notions of time. Students submit their assignments up to the last minute of the day; educational processes continue 24/7.</td>
</tr>
<tr>
<td>16</td>
<td>C</td>
<td>Initiative</td>
<td>PRE_SS_F_3</td>
<td>[ProdDesc] Course planning, logistics and study support</td>
<td>EARLY_BP_3</td>
<td>The Roster is not filled up completely and consistently by teachers.</td>
</tr>
<tr>
<td>17</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_SS_F_2</td>
<td>[ProdDesc] The workspace function of TeleTOP is improved. It can now allow submission of large files in the Workspace, i.e. 1 MB or more.</td>
<td>EARLY_BP_2</td>
<td>Students should not be able to copy each other’s work or look into other group’s solution.</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_[RFP]*</td>
<td>All statements in the RFP listed in early implementation.</td>
<td>EARLY_BP_2</td>
<td>LMS Supplier who responded to the RFP was too expensive.</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_[RFP]*</td>
<td>All statements in the RFP listed in early implementation.</td>
<td>POST_BP_2</td>
<td>Training session set-up is increasingly becoming tiresome. Oracle iLearning performance issues are becoming more prevalent. Downtimes are getting frequent; there is almost no training session setup with no down time or error messages received while uploading new materials.</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_SS_F_3</td>
<td>Oracle iLearning should support multiple curricula, i.e. it should be possible to incorporate Powerpoint slides and exercises in pdf format used in classroom-based training.</td>
<td>POST_BP_4</td>
<td>Quality of training deteriorates due to time outs during onsite training. Trainer reports that the course evaluation takes too long to load and often leads to time outs.</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SS_F_16</td>
<td>[RFP8] Ability to attach PowerPoint and Word Documents as reference material.</td>
<td>POST_BP_3</td>
<td>Negative course feedback received as trainee access to learning content, i.e. a pdf file or slide is getting slower.</td>
</tr>
<tr>
<td>22</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_[RFP]*</td>
<td>All statements in the RFP listed in early implementation.</td>
<td>POST_BP_6</td>
<td>Newer members of the team, i.e. the new training developer finds Oracle iLearning a difficult system to use. Even after having an intensive orientation, she finds training setup tasks in Oracle to be too complicated.</td>
</tr>
<tr>
<td>23</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SS_F_19</td>
<td>[RFP12] Assessment Utility for creating scored exams.</td>
<td>POST_BP_7</td>
<td>After so many tries, the new training developer cannot get an assessment up and running in Oracle iLearning.</td>
</tr>
<tr>
<td>24</td>
<td>D</td>
<td>Initiative</td>
<td>EARLY_SS_F_19</td>
<td>[RFP12] Assessment Utility for creating scored exams.</td>
<td>POST_BP_8</td>
<td>User help and how-to tutorials, especially in setting up assessments, seem to be unavailable.</td>
</tr>
<tr>
<td>25</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_SS_F_1</td>
<td>User subscription should be increased to 135 users.</td>
<td>POST_BP_5</td>
<td>New FDS training manager wants to cut down costs.</td>
</tr>
</tbody>
</table>
M7. Business problem resolved by (re)defining software product concept

<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
<td>POST_SP_1</td>
<td>KENNISNET should be converted into a project management system.</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The KCS wants the revised version of KENNISNET to be intensively used by the non-life insurance experts.</td>
<td>POST_SP_2</td>
<td>KENNISNET should be in the form of an expert system.</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_BP_1</td>
<td>The institute wants to participate in the renewal and modernization efforts in the educational sector through ICT.</td>
<td>PRE_SP_1</td>
<td>The software has to be an e-learning software.</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_BP_2</td>
<td>The institute wants to implement an e-learning software.</td>
<td>PRE_SP_1</td>
<td>The software has to be an e-learning software.</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_BP_3</td>
<td>IT Department and FCC administrators at OI. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.</td>
<td>PRE_SP_1</td>
<td>The software has to be an e-learning software.</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>Initiative</td>
<td>PRE_BP_3</td>
<td>IT Department and FCC administrators at OI. Their goal is to innovate educational and administrative processes in the institute by finding and implementing ICT solutions.</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.</td>
<td>EARLY_SP_1</td>
<td>The COTS software FCC is also communication and collaboration platform.</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>The way in which teacher-trainers gather, develop and exchange educational materials need to be improved. The process has to be carried out in a more efficient as well cost and time effective manner.</td>
<td>POST_SP_3</td>
<td>FCC supports e-learning and distance learning.</td>
</tr>
<tr>
<td>9</td>
<td>B</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>The teacher-trainers want to broaden their about the useful functions of the software in addition to the ones that they already know.</td>
<td>POST_SP_4</td>
<td>FCC serves as a ‘closed’ system exclusively accessible only to OI employees.</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_BP_1</td>
<td>Students should submit their assignments on time or ahead of time. The continued availability of TeleTOP blurs the official notions of time. Students submit their assignments up to the last minute of the day; educational processes continue 24/7.</td>
<td>EARLY_BS_4</td>
<td>Students should submit their assignments by hand. If agreed to be submitted online, it should be uploaded in TeleTOP at certain time period, i.e. 17:30</td>
</tr>
<tr>
<td>11</td>
<td>C</td>
<td>Breakdown</td>
<td>EARLY_BP_3</td>
<td>The Roster is not filled up completely and consistently by teachers.</td>
<td>EARLY_BS_6</td>
<td>Make use of a student assistants to aide teachers in setting up TeleTOP sites for the course</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>Breakdown</td>
<td>PRE_BP_6</td>
<td>FDS feels the need to bring (new) employees up to speed in terms of knowledge and competency.</td>
<td>EARLY_SP_1</td>
<td>FDS Academy, tasked to develop the program, would like to make use of ICT solutions such as e-learning and learning management systems (LMS) that could help in reducing induction period for employees.</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>Breakdown</td>
<td>EARLY_BP_2</td>
<td>LMS Supplier who responded to the RFP was too expensive.</td>
<td>EARLY_SP_2</td>
<td>Oracle iLearning proposed as affordable and suitable LMS alternative; FDS Academy</td>
</tr>
</tbody>
</table>

7 For Case A, this is added or updated after the results of the second case, see Chapter 5, section 5.7.2
<table>
<thead>
<tr>
<th>Item (#)</th>
<th>Case</th>
<th>Trigger</th>
<th>Source (R1)</th>
<th>Requirements Statement</th>
<th>Update (R2)</th>
<th>Requirements Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_1</td>
<td>Increasingly, Sales and US-based account managers continue to request for onsite training to customers who didn’t sign a software license agreement yet. Expressed knowledge gap by customers on FDS software is seen as a training need.</td>
<td>POST_SP_1</td>
<td>Oracle iLearning should be seen as a sales and account management support tool.</td>
</tr>
<tr>
<td>15</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_5</td>
<td>New FDS training manager wants to cut down costs.</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>16</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_2</td>
<td>Training session set-up is increasingly becoming tiresome. Oracle iLearning performance issues are becoming more prevalent. Downtimes are getting frequent there is almost no training session setup with no down time or error messages received while uploading new materials.</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>17</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_3</td>
<td>Negative course feedback received as trainee access to learning content, i.e. a pdf file or slide is getting slower.</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_4</td>
<td>Quality of training deteriorates due to time outs during onsite training. Trainer reports that the course evaluation takes too long to load and often leads to time outs.</td>
<td>POST_SP_2</td>
<td>Course evaluation doesn’t have to take place in Oracle iLearning anymore.</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_5</td>
<td>Training coordinator would like to migrate course evaluation form after seeing the Marketing Department’s online survey environment. New FDS training manager wants to cut down costs.</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_7</td>
<td>After so many tires, the new training developer cannot get an assessment up and running in Oracle iLearning.</td>
<td>POST_SP_5</td>
<td>Other e-learning authoring tool, i.e. Adobe Captivate should be used for creating assessments instead of Oracle iLearning.</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_7</td>
<td>After so many tires, the new training developer cannot get an assessment up and running in Oracle iLearning.</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>22</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_6</td>
<td>Newer members of the team, i.e. the new training developer finds Oracle iLearning a difficult system to use. Even after having an intensive orientation, she finds training setup tasks in Oracle to be too complicated.</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>23</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_8</td>
<td>User help and how-to tutorials, especially in setting up assessments, seem to be unavailable.</td>
<td>POST_SP_3</td>
<td>Oracle iLearning should cost less and must be replaced by an open source LMS.</td>
</tr>
<tr>
<td>24</td>
<td>D</td>
<td>Breakdown</td>
<td>POST_BP_10</td>
<td>FDS is not able to sign any license agreement in the last 6 months of 2008. The prognosis for next year is zero sales. Customers are cancelling maintenance and support agreements.</td>
<td>POST_SP_4</td>
<td>With the dissolution of the training department, Oracle iLearning is no longer needed. Subscription has to be at minimum to meet contractual obligation with customers.</td>
</tr>
</tbody>
</table>


Hillebrand, F., Trosby, F., Holley, K., & Harris, I. (2010). *Short Message Service (SMS): The Creation of Personal Global Text Messaging*. Chichester, West Sussex: John Wiley & Sons Ltd.


Jesitus, J. (1997). Broken Promises? FoxMeyer's Project was a Disaster. Was the Company Too Aggressive or was it Misled? Industry Week, 31-37.


Orlikowski, W. J. (1993). *CASE tools as organizational change: investigating incremental and radical changes in systems development*. Massachusetts Institute of Technology (MIT), Sloan School of Management.


Heath.


SAMENVATTING

Het is algemeen bekend dat bij software ontwikkeling de requirements zich voortdurend wijzigen. Ook is bekend dat gedurende de hele levenscyclus van een systeem de requirements blijven veranderen. Toch is het evolueren van requirements een slecht begrepen fenomeen. De meeste studies over zich evoluïerende requirements richten zich op veranderingen in geschreven specificaties en op software-architectuur en ontwerp. Meestal ligt de nadruk op de fase van de software ontwikkeling. Er is weinig bekend over de manier waarop de requirements zich evolueren nadat de software in gebruik wordt genomen.

Een voorbeeld van toepassingen waarbij de requirements zich blijven ontwikkelen nadat het systeem wordt ingezet in de organisatie is ‘groupware’. Dat zijn ICT (software + hardware) toepassingen die de coöperatieve processen van individuen in groepen ondersteunen. In moderne zakelijke toepassingen en grote informatiesystemen is steeds vaker groupware functionaliteit aanwezig. Coöperatieve processen ondersteund door groupware-toepassingen hebben geen heldere structuur. Integendeel, de structuur evolueert zich vaak spontaan en kan niet eenvoudig worden gespecificeerd. Het is moeilijk te voorzien hoe een groupware systeem later in de operationele omgeving gebruikt gaat worden. De ongrijpbare natuur van menselijke groepsprocessen maakt het moeilijk om groupware specificaties te genereren.

Groupware technologie heeft – als software voor de ondersteuning van de coöperatieve processen van samenwerkende mensen - het potentieel om ingrijpende veranderingen in organisaties te bewerken. Diverse studies van groupware implementaties wijzen op veranderende organisatorische behoeften. De interactie tussen gebruikers en software leidt tot verbeteringen in de prestatie, nieuwe vormen van communicatie, veranderingen in de groepsstructuur en groepsfunctioneren. Dit maakt duidelijk dat de eisen zijn gewijzigd.

Deze studie is een empirisch onderzoek naar de requirements evolutie van bestaande groupware systemen door middel van case studies. Het doel is bij te dragen tot de ontwikkeling van een theorie van zich evoluïerende specificaties. Om richting en structuur te geven aan het onderzoek is een conceptueel raamwerk ontwikkeld dat een geïntegreerd overzicht van requirements als een matrix van domeinen geeft. Het concept neemt de brede dimensies van het bedrijfsleven, software, problemen en oplossingen als eisen die aanleiding geven tot vier domeinen van requirements: business problem, business solution, software product concept, en software solution specification. Gestart is met requirements evolutie geformuleerd als de verandering in de requirements over de tijd.

Het toepassingsgebied van groupware werd gekozen als de empirische omgeving voor de observatie van veranderingen in de requirements gedurende het gebruik van het systeem. Er zijn vier case studies van groupware implementaties uitgevoerd. Twee mislukte implementaties en twee succesvolle implementaties werden onderzocht. Het conceptueel raamwerk wordt gebruikt om de cases te analyseren en is geactualiseerd en verbeterd op basis van een evaluatie over hoe het bijgedragen heeft in het verwerven van inzicht in requirements evolutie. Een definitieve versie van het conceptuele kader is ontwikkeld en dit is gebruikt om de laatste twee case studies te analyseren.

De resultaten bevatten de ontdekking van het verband tussen de verschillende requirements domeinen. Het verband zijn de vaak terugkerende mechanismen waarbij veranderde en nieuwe eisen leiden tot nieuwe eisen op verschillende domeinen. Uiteindelijk wordt requirements
evolutie uitgedrukt in de resolutie van een *breakdown* of een initiatief in een overzicht van de meest invloedrijke betrekkingen. De belangrijkste bijdrage van dit onderzoek is een overzicht van requirements evolutie patronen: samenvoeging van zich opeenvolgende invloedrijke betrekkingen die de pijnlijk bekende patronen van gedrag in systeem implementaties verklaren.
SIKS Dissertatieruks

---------------
2009
---------------

2009-01
Rasa Jurgenaita (RUN)
Symmetric Causal Independence Models

2009-02
Willem Robert van Hage (VU)
Evaluating Ontology-Alignment Techniques

2009-03
Hans Stol (UvT)
A Framework for Evidence-based Policy Making Using IT

2009-04
Josephine Nabukonya (RUN)
Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-05
Sietse Overbeek (RUN)
Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality

2009-06
Muhammad Subianto (UU)
Understanding Classification

2009-07
Ronald Poppe (UT)
Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-08
Volker Nannen (VU)
Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-09
Benjamin Kanagwa (RUN)
Design, Discovery and Construction of Service-oriented Systems

2009-10
Jan Wielemaker (UVA)
Logic programming for knowledge-intensive interactive applications

2009-11
Alexander Boer (UVA)
Legal Theory, Sources of Law & the Semantic Web

2009-12
Peter Massu (TUE, Humboldt-Universitaet zu Berlin)
Operating Guidelines for Services

2009-13
Steven de Jong (UM)
Fairness in Multi-Agent Systems
2009-14
Maksym Korotkiy (VU)
From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15
Rinke Hoekstra (UVA)
Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16
Fritz Reul (UvT)
New Architectures in Computer Chess

2009-17
Laurens van der Maaten (UvT)
Feature Extraction from Visual Data

2009-18
Fabian Groffen (CWI)
Armada, An Evolving Database System

2009-19
Valentin Robu (CWI)
Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets

2009-20
Bob van der Vecht (UU)
Adjustable Autonomy: Controlling Influences on Decision Making

2009-21
Stijn Vanderlooy (UM)
Ranking and Reliable Classification

2009-22
Pavel Serdyukov (UT)
Search For Expertise: Going beyond direct evidence

2009-23
Peter Hofgesang (VU)
Modelling Web Usage in a Changing Environment

2009-24
Annerieke Heuvelink (VUA)
Cognitive Models for Training Simulations

2009-25
Alex van Ballegooij (CWI)
"RAM: Array Database Management through Relational Mapping"

2009-26
Fernando Koch (UU)
An Agent-Based Model for the Development of Intelligent Mobile Services

2009-27
Christian Glahn (OU)
Contextual Support of social Engagement and Reflection on the Web

2009-28
Sander Evers (UT)
Sensor Data Management with Probabilistic Models
2009-29
Stanislav Pokraev (UT)
Model-Driven Semantic Integration of Service-Oriented Applications

2009-30
Marcin Zukowski (CWI)
Balancing vectorized query execution with bandwidth-optimized storage

2009-31
Sofiya Katrenko (UVA)
A Closer Look at Learning Relations from Text

2009-32
Rik Farenhorst (VU) and Remco de Boer (VU)
Architectural Knowledge Management: Supporting Architects and Auditors

2009-33
Khiet Truong (UT)
How Does Real Affect Affect Recognition In Speech?

2009-34
Inge van de Weerd (UU)
Advancing in Software Product Management: An Incremental Method Engineering Approach

2009-35
Wouter Koelewijn (UL)
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling

2009-36
Marco Kalz (OUN)
Placement Support for Learners in Learning Networks

2009-37
Hendrik Drachslcr (OUN)
Navigation Support for Learners in Informal Learning Networks

2009-38
Riina Vuorikari (OU)
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context

2009-39
Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)
Service Substitution -- A Behavioral Approach Based on Petri Nets

2009-40
Stephan Raaijmakers (UvT)
Multinomial Language Learning: Investigations into the Geometry of Language

2009-41
Igor Berezhnyy (UvT)
Digital Analysis of Paintings

2009-42
Toine Bogers (UvT)
Recommender Systems for Social Bookmarking

2009-43
Virginia Nunes Leal Franqueira (UT)
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients

2009-44
Roberto Santana Tapia (UT)
Assessing Business-IT Alignment in Networked Organizations
2009-45
  Jilles Vreeken (UU)
  Making Pattern Mining Useful

2009-46
  Loredana Afanasiev (UvA)
  Querying XML: Benchmarks and Recursion

2010-01
  Matthijs van Leeuwen (UU)
  Patterns that Matter

2010-02
  Ingo Wassink (UT)
  Work flows in Life Science

2010-03
  Joost Geurts (CWI)
  A Document Engineering Model and Processing Framework for Multimedia documents

2010-04
  Olga Kulyk (UT)
  Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments

2010-05
  Claudia Hauff (UT)
  Predicting the Effectiveness of Queries and Retrieval Systems

2010-06
  Sander Bakkes (UvT)
  Rapid Adaptation of Video Game AI

2010-07
  Wim Fikkert (UT)
  Gesture interaction at a Distance

2010-08
  Krzysztof Siewicz (UL)
  Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments

2010-09
  Hugo Kielman (UL)
  A Politie gegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10
  Rebecca Ong (UL)
  Mobile Communication and Protection of Children

2010-11
  Adriaan Ter Mors (TUD)
  The world according to MARP: Multi-Agent Route Planning

2010-12
  Susan van den Braak (UU)
  Sensemaking software for crime analysis
2010-13
  Gianluigi Folino (RUN)
  High Performance Data Mining using Bio-inspired techniques

2010-14
  Sander van Splunter (VU)
  Automated Web Service Reconfiguration

2010-15
  Lianne Bodenstaff (UT)
  Managing Dependency Relations in Inter-Organizational Models

2010-16
  Sicco Verwer (TUD)
  Efficient Identification of Timed Automata, theory and practice

2010-17
  Spyros Kotoulas (VU)
  Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18
  Charlotte Gerritsen (VU)
  Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19
  Henriette Cramer (UvA)
  People's Responses to Autonomous and Adaptive Systems

2010-20
  Ivo Swartjes (UT)
  Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21
  Harold van Heerde (UT)
  Privacy-aware data management by means of data degradation

2010-22
  Michiel Hildebrand (CWI)
  End-user Support for Access to Heterogeneous Linked Data

2010-23
  Bas Steunebrink (UU)
  The Logical Structure of Emotions

2010-24
  Dmytro Tykhonov
  Designing Generic and Efficient Negotiation Strategies

2010-25
  Zulfikar Ali Memon (VU)
  Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26
  Ying Zhang (CWI)
  XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27
  Marten Voulon (UL)
  Automatisch contracteren

2010-28
  Arne Koopman (UU)
  Characteristic Relational Patterns
2010-29
Stratos Idreos(CWI)
Database Cracking: Towards Auto-tuning Database Kernels

2010-30
Marieke van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31
Victor de Boer (UVA)
Ontology Enrichment from Heterogeneous Sources on the Web

2010-32
Marcel Hiel (UvT)
An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems

2010-33
Robin Aly (UT)
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34
Teduh Dirgahayu (UT)
Interaction Design in Service Compositions

2010-35
Dolf Trieschnigg (UT)
Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36
Jose Janssen (OU)
Paving the Way for Lifelong Learning: Facilitating competence development through a learning path specification

2010-37
Niels Lohmann (TUE)
Correctness of services and their composition

2010-38
Dirk Fahland (TUE)
From Scenarios to components

2010-39
Ghazanfar Farooq Siddiqui (VU)
Integrative modeling of emotions in virtual agents

2010-40
Mark van Assem (VU)
Converting and Integrating Vocabularies for the Semantic Web

2010-41
Guillaume Chaslot (UM)
Monte-Carlo Tree Search

2010-42
Sybren de Kinderen (VU)
Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach

2010-43
Peter van Kranenburg (UU)
A Computational Approach to Content-Based Retrieval of Folk Song Melodies

2010-44
Pieter Bellekens (TUE)
An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45
Vasilios Andrikopoulos (UvT)
A theory and model for the evolution of software services

2010-46
Vincent Pijpers (VU)
e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47
Chen Li (UT)
Mining Process Model Variants: Challenges, Techniques, Examples

2010-48
Withdrawn

2010-49
Jahn-Takeshi Saito (UM)
Solving difficult game positions

2010-50
Bouke Huurnink (UVA)
Search in Audiovisual Broadcast Archives

2010-51
Alia Khairia Amin (CWI)
Understanding and supporting information seeking tasks in multiple sources

2010-52
Peter-Paul van Maanen (VU)
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention

2010-53
Edgar Meij (UVA)
Combining Concepts and Language Models for Information Access

======

2011

======

2011-01
Botond Cseke (RUN)
Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02
Nick Tinnemeier(UU)
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03
Jan Martijn van der Werf (TUE)
Compositional Design and Verification of Component-Based Information Systems

2011-04
Hado van Hasselt (UU)
Insights in Reinforcement Learning: Formal analysis and empirical evaluation of temporal-difference learning algorithms
2011-05
Base van der Raadt (VU)
Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06
Yiwen Wang (TUE)
Semantically-Enhanced Recommendations in Cultural Heritage

2011-07
Yujia Cao (UT)
Multimodal Information Presentation for High Load Human Computer Interaction

2011-08
Nieske Vergunst (UU)
BDI-based Generation of Robust Task-Oriented Dialogues

2011-09
Tim de Jong (OU)
Contextualised Mobile Media for Learning

2011-10
Bart Bogaert (UvT)
Cloud Content Contention

2011-11
Dhaval Vyas (UT)
Designing for Awareness: An Experience-focused HCI Perspective

2011-12
Carmen Bratosin (TUE)
Grid Architecture for Distributed Process Mining

2011-13
Xiaoyu Mao (UvT)
Airport under Control. Multiagent Scheduling for Airport Ground Handling

2011-14
Milan Lovric (EUR)
Behavioral Finance and Agent-Based Artificial Markets

2011-15
Marijn Koolen (UvA)
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

2011-16
Maarten Schadd (UM)
Selective Search in Games of Different Complexity

2011-17
Jiyin He (UVA)
Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18
Mark Ponsen (UM)
Strategic Decision-Making in complex games

2011-19
Ellen Rusman (OU)
The Mind ’ s Eye on Personal Profiles

2011-20
Qing Gu (VU)
Guiding service-oriented software engineering - A view-based approach
2011-21
Linda Terlouw (TUD)
Modularization and Specification of Service-Oriented Systems

2011-22
Junte Zhang (UVA)
System Evaluation of Archival Description and Access

2011-23
Wouter Weerkamp (UVA)
Finding People and their Utterances in Social Media

2011-24
Herwin van Welbergen (UT)
Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

2011-25
Syed Waqar ul Qounain Jaffry (VU))
Analysis and Validation of Models for Trust Dynamics

2011-26
Matthijs Aart Pontier (VU)
Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

2011-27
Aneel Bhulai (VU)
Dynamic website optimization through autonomous management of design patterns

2011-28
Rianne Kaptein (UVA)
Effective Focused Retrieval by Exploiting Query Context and Document Structure

2011-29
Faisal Kamiran (TUE)
Discrimination-aware Classification

2011-30
Egon van den Broek (UT)
Affective Signal Processing (ASP): Unraveling the mystery of emotions

2011-31
Ludo Waltman (EUR)
Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

2011-32
Nees-Jan van Eck (EUR)
Methodological Advances in Bibliometric Mapping of Science

2011-33
Tom van der Weide (UU)
Arguing to Motivate Decisions

2011-34
Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35
Maaike Harbers (UU)
Explaining Agent Behavior in Virtual Training
2011-36
Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach

2011-37
Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference

2011-38
Nyree Lemmens (UM)
Bee-inspired Distributed Optimization

2011-39
Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games

2011-40
Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development

2011-41
Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control

2011-42
Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution

2011-43
Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge

2011-44
Boris Reuderink (UT)
Robust Brain-Computer Interfaces

2011-45
Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection

2011-46
Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work

2011-47
Azizi Bin Ab Aziz(VU)
Exploring Computational Models for Intelligent Support of Persons with Depression

2011-48
Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent

2011-49
Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality
2012-01
Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda

2012-02
Muhammad Umair (VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-03
Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories

2012-04
Jurriaan Souer (UU)
Development of Content Management System-based Web Applications

2012-05
Marijn Plomp (UU)
Maturing Interorganisational Information Systems

2012-06
Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks

2012-07
Rianne van Lamalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions

2012-08
Gerben de Vries (UVA)
Kernel Methods for Vessel Trajectories

2012-09
Ricardo Neisse (UT)
Trust and Privacy Management Support for Context-Aware Service Platforms

2012-10
David Smits (TUE)
Towards a Generic Distributed Adaptive Hypermedia Environment

2012-11
J.C.B. Rantham Prabhakara (TUE)
Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12
Kees van der Sluijs (TUE)
Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13
Suleman Shahid (UvT)
Fun and Face: Exploring non-verbal expressions of emotion during playful interactions

2012-14
Evgeny Knutov (TUE)
Generic Adaptation Framework for Unifying Adaptive Web-based Systems
2012-15
Natalie van der Wal (VU)
Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.

2012-16
Fienke Both (VU)
Helping people by understanding them - Ambient Agents supporting task execution and depression treatment

2012-17
Amal Elgammal (UvT)
Towards a Comprehensive Framework for Business Process Compliance

2012-18
Eltjo Poort (VU)
Improving Solution Architecting Practices

2012-19
Helen Schonenberg (TUE)
What's Next? Operational Support for Business Process Execution

2012-20
Ali Bahramisharif (RUN)
Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing

2012-21
Roberto Cornacchia (TUD)
Querying Sparse Matrices for Information Retrieval

2012-22
Thijs Vis (UvT)
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?

2012-23
Christian Muehl (UT)
Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction

2012-24
Laurens van der Werff (UT)
Evaluation of Noisy Transcripts for Spoken Document Retrieval

2012-25
Silja Eckartz (UT)
Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application

2012-26
Emile de Maat (UVA)
Making Sense of Legal Text

2012-27
Hayrettin Gurkok (UT)
Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games

2012-28
Nancy Pascale (UvT)
Engendering Technology Empowering Women

2012-29
Almer Tigelaar (UT)
Peer-to-Peer Information Retrieval
2012-30
Alina Pommeranz (TUD)
Designing Human-Centered Systems for Reflective Decision Making

2012-31
Emily Bagarukayo (RUN)
A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure

2012-32
Wietske Visser (TUD)
Qualitative multi-criteria preference representation and reasoning

2012-33
Rory Sie (OUN)
Coalitions in Cooperation Networks (COCOON)

2012-34
Pavol Jancura (RUN)
Evolutionary analysis in PPI networks and applications

2012-35
Evert Haasdijk (VU)
Never Too Old To Learn -- On-line Evolution of Controllers in Swarm- and Modular Robotics

2012-36
Denis Ssebugwawo (RUN)
Analysis and Evaluation of Collaborative Modeling Processes

2012-37
Agnes Nakakawa (RUN)
A Collaboration Process for Enterprise Architecture Creation

2012-38
Selmar Smit (VU)
Parameter Tuning and Scientific Testing in Evolutionary Algorithms

2012-39
Hassan Fatemi (UT)
Risk-aware design of value and coordination networks

2012-40
Agus Gunawan (UvT)
Information Access for SMEs in Indonesia

2012-41
Sebastian Kelle (OU)
Game Design Patterns for Learning

2012-42
Dominique Verpoorten (OU)
Reflection Amplifiers in self-regulated Learning

2012-43
Withdrawn

2012-44
Anna Tordai (VU)
On Combining Alignment Techniques

2012-45
Benedikt Kratz (UvT)
A Model and Language for Business-aware Transactions
2012-46
Simon Carter (UVA)
Exploration and Exploitation of Multilingual Data for Statistical Machine Translation

2012-47
Manos Tsagkias (UVA)
Mining Social Media: Tracking Content and Predicting Behavior

2012-48
Jorn Bakker (TUE)
Handling Abrupt Changes in Evolving Time-series Data

2012-49
Michael Kaisers (UM)
Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions

======
2013
======

2013-01
Viorel Milea (EUR)
News Analytics for Financial Decision Support

2013-02
Erietta Liarou (CWI)
MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing

2013-03
Szymon Klarman (VU)
Reasoning with Contexts in Description Logics

2013-04
Chetan Yadati (TUD)
Coordinating autonomous planning and scheduling

2013-05
Dulce Pumareja (UT)
Groupware Requirements Evolution Patterns