

Lappeenrannan teknillinen yliopisto
Lappeenranta University of Technology

Uolevi Nikula

**INTRODUCING BASIC SYSTEMATIC REQUIREMENTS
ENGINEERING PRACTICES IN SMALL ORGANIZATIONS
WITH AN EASY TO ADOPT METHOD**

*Thesis for the degree of Doctor of Science (Technology) to
be presented with due permission for public examination
and criticism in the Auditorium of the Student Union
House at Lappeenranta University of Technology,
Finland, on the 5th of November, 2004, at noon.*

Supervisor Professor Heikki Kälviäinen
Department of Information Technology
Lappeenranta University of Technology
Finland

Reviewers Professor Ilkka Haikala
Department of Information Technology
Tampere University of Technology
Finland

Professor Juhani Iivari
Department of Information Processing Science
University of Oulu
Finland

Opponent Professor Donald C. Gause
Department of Bioengineering
Binghamton University
Binghamton, New York, USA

ISBN 951-764-948-7
ISBN 951-764-952-5 (PDF)
ISSN 1456-4491

Lappeenrannan teknillinen yliopisto
Digipaino 2004

ABSTRACT

Uolevi Nikula

Introducing Basic Systematic Requirements Engineering Practices in Small Organizations with an Easy to Adopt Method

Lappeenranta 2004

207 p.

Acta Universitatis Lappeenrantaensis 189

Diss. Lappeenranta University of Technology

ISBN 951-764-948-7, ISSN 1456-4491

Requirements-related issues have been found the third most important risk factor in software projects and as the biggest reason for software project failures. This is not a surprise since requirements engineering (RE) practices have been reported deficient in more than 75% of all enterprises. A problem analysis on small and low maturity software organizations revealed two central reasons for not starting process improvement efforts: lack of resources and uncertainty about process improvement effort paybacks.

In the constructive part of the study a basic RE method, BaRE, was developed to provide an easy to adopt way to introduce basic systematic RE practices in small and low maturity organizations. Based on diffusion of innovations literature, thirteen desirable characteristics were identified for the solution and the method was implemented in five key components: requirements document template, requirements development practices, requirements management practices, tool support for requirements management, and training.

The empirical evaluation of the BaRE method was conducted in three industrial case studies. In this evaluation, two companies established a completely new RE infrastructure following the suggested practices while the third company conducted continued requirements document template development based on the provided template and used it extensively in practice. The real benefits of the adoption of the method were visible in the companies in four to six months from the start of the evaluation project, and the two small companies in the project completed their improvement efforts with an input equal to about one person month. The collected data on the case studies indicates that the companies implemented new practices with little adaptations and little effort. Thus it can be concluded that the constructed BaRE method is indeed easy to adopt and it can help introduce basic systematic RE practices in small organizations.

Keywords: requirements engineering, software engineering, method engineering, agile methods, domain engineering, software process improvement, technology transfer, diffusion of innovations

UDC 004.414.3 : 004.413

ACKNOWLEDGEMENTS

This thesis was developed under the East Finland Graduate School in Computer Science and Engineering (ECSE). This arrangement made it possible to appoint Professor Jorma Sajaniemi from the University of Joensuu as my supervisor to support Professor Heikki Kälviäinen from Lappeenranta University of Technology in this work. I wish to thank both my supervisors for their support in the course of this project. Professor Sajaniemi presented assiduous support by reading many versions of my many documents and provided invaluable comments on them. He also served as the methodological backbone of the work and knew the right answers when skepticism started to threaten the progress of the study. Professor Kälviäinen was instrumental in handling the administrative parts of the study and he also helped in getting the details right in this thesis.

The co-operation with the University of Joensuu was essential also in the practical aspects of the work, since the industry partners for the empirical evaluation came from Joensuu. This co-operation was implemented under the technology transfer project tSoft at the University of Joensuu. I am very grateful to the three companies for the opportunity to work with them. I am also indebted to 13 other companies with which I have co-operated during my work on this thesis.

I wish to express my gratitude to the Department of Information Technology for the possibility of doing this thesis as a researcher in the department. The Department of Computer Science at the University of Joensuu, and especially Professor Jussi Parkkinen and Professor Markku Tukiainen, the leader of the tSoft project, provided important support for this project with their willingness to participate in the project costs in the Joensuu area.

An earlier version of this thesis was published as my licentiate thesis. Both the licentiate thesis and the current work have been reviewed by the same external reviewers, Professor Ilkka Haikala from Tampere University of Technology and Professor Juhani Iivari from the University of Oulu. I am very grateful to them both for the time they devoted to my theses and the insightful comments they provided on them. Professor Ilkka Haikala played a critical role also in the beginning of my researcher career since he helped me to get started with my graduate studies and demonstrated a natural way of doing research to which I have tried to stay true. Professor Juhani Iivari, on the other hand, was more than generous in devoting his time to help me finalize my thesis as a piece of scientific work. His seemingly endless knowledge of research papers focusing on the vaguely addressed topics in earlier versions of this thesis showed in practice the power and elegance of the scientific approach; I hope my future research efforts will better demonstrate the adoption of this scientific way of working.

My most important colleagues at Lappeenranta University of Technology during the present study were Kari Smolander and Päivi Ovaska, to whom I am grateful for many insightful discussions on conducting research. From the University of Joensuu especially Ilmari Saastamoinen is acknowledged for his help in tackling the SPICE related issues. In the course of the present study I have also been in contact with many established researchers who have generously devoted their time and encouragement to this effort and I want to express my gratitude especially to Kalle Lyytinen, Richard L. Baskerville, Olly Gotel, Pete Sawyer, Benjamin L. Kovitz, and Andrew Vickers. Even though their input may have appeared

insignificant, from the point of view of a beginning researcher from the middle of nowhere things look different.

The direct financial support from Lappeenrannan teknillisen yliopiston tukisäätiö, The Finnish Foundation for Economic and Technology Sciences – KAUTE –, Teknologian Edistämissäätiö, Ulla Tuomisen säätiö, and Itä-Suomen korkean teknologian säätiö is also gratefully acknowledged.

Finally, I want to thank my parents, my mother-in-law, my wife Minna, and Leevi and Aaron for bearing with an all too often absentminded and busy researcher.

Lappeenranta, October 2004

Uolevi Nikula

*To the young explorers Leevi and Aaron and their guide Minna
who continuously demonstrate
the joy of experimenting and discovering new things.*

PROLOGUE

After getting acquainted with the wealth of the research in requirements engineering, a feeling of uncertainty crept up in the mind of a young researcher. Luckily reading yet another paper confirmed the observation – I was not alone. This paper, an introduction to a conference mini-tutorial in technology transfer, described the situation between research and practice as follows (Greenspan 1998):

In the spring of each year, the two friends made a special point of getting together for a drink during the annual software engineering conference. “Requirements engineering,” the researcher insisted, for the n^{th} year in a row, “is the answer to your problems!” He lifted the tall glass of pilsner, took a sip, and looked across the table at his colleague for recognition of the obvious.

“I told you, we don’t have time for that,” clipped the software development manager, sensitive as usual about this subject. ... [A year goes by]

“Well, I mentioned the idea [of RE] to my management,” admitted the manager, “but they said that what we need to engineer are software systems, not requirements. I didn’t know how to respond to that.”

“Tell them that every dollar they spend on Requirements Engineering will be recovered later – tenfold, maybe a hundredfold. Tell them...” [A year goes by]

“Okay. I’ve convinced the people back at home that our problem is poor attention to requirements, and that the solution is to make Requirements Engineering a top priority. I have just one question for you: What do we do now?”

A long silence followed. The researcher stared into his teacup for inspiration.

“Hmm...I don’t know. How about if we talk about that next year.”

LIST OF ABBREVIATIONS

(e)	Estimated
ABA	Administrative and Business Applications
ACM	The Association of Computing Machinery
A1-2, B1-3, C1-3	Key participants in the evaluation project formed from case identifier and person number
BaRE	Basic Requirements Engineering
CMM	Capability Maturity Model
COTS	Commercial-Off-The-Shelf
CRUD	Create, Read, Update, and Delete operations
DB	Database
Desn	Designed
DSDM	Dynamic Systems Development
eng.	engineering
GQM	Goal Question Metric
ICSE	International Conference on Software Engineering
IEEE	The Institute of Electrical and Electronics Engineers
Impl	Implemented
IS	Information System
IT	Information Technology
kEUR	kilo/thousand euros
ME	Method Engineering
MO	Month
MS	Microsoft
P0, P1, P2, P3	Phase 0, 1, and 2 of the BaRE evaluation project; P0 refers to the time before the project, 1 to the first and 2 to the second half of it
PM	Person Month
PSP	Personal Software Process
PW	Person Week
QFD	Quality Function Deployment
RD	Requirements Document
RDev	Requirements Development
RE	Requirements Engineering
REAIMS	Requirements Engineering Adaptation and Improvement for Safety and dependability
Rel	Released
RM	Requirements Management
RUP	Rational Unified Process
SEI	The Software Engineering Institute at Carnegie Mellon University, USA
SIM	Subscriber Identification Module

Spec	Specified
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability dEtermination
SME	Small and Medium Enterprise
SREM	Software Requirements Engineering Methodology
SRS	Software Requirements Specification
SSM	Soft Systems Methodology
SW	Software
TSP	Team Software Process
tSoft	A software engineering technology transfer program at the University of Joensuu, Finland
TTM	Time-To-Market
UI	User Interface
UK	United Kingdom
UML	Unified Modeling Language
US	United States (of America)
VORD	Viewpoint Oriented Requirements Definition
VP	Viewpoint
XP	Extreme Programming

TABLE OF CONTENTS

1. INTRODUCTION.....	15
1.1. Basic Question of the Thesis	15
1.2. Research Questions	16
1.3. Research Methods	17
1.3.1. Constructive Research.....	17
1.3.2. Case Study Research.....	20
1.4. Research Contribution.....	21
1.5. Emergence of the Thesis	22
1.6. Structure of the Thesis.....	25
2. TERMS AND DEFINITIONS.....	26
2.1. Context	26
2.2. Requirements Engineering	27
2.3. Requirements Documentation	31
2.4. Methods.....	32
3. LITERATURE SURVEY	37
3.1. Software Projects and Requirements.....	37
3.1.1. The CHAOS Report	37
3.1.2. Software Project Risks	40
3.1.3. Software Project Success Factors	41
3.1.4. Summary.....	42
3.2. Research Areas of Interest.....	43
3.2.1. Requirements Engineering	44
3.2.2. Agile Software Development	49
3.2.3. Domain Engineering	51
3.2.4. Method Engineering.....	52
3.2.5. Technology Transfer	55
3.2.6. Process Improvement.....	60
3.2.7. Summary.....	66
3.3. Technical Issues	67
3.3.1. Central Elements of Methods	68
3.3.2. Key Issues in RE Tool and Technique Selection.....	70
3.3.3. Summary.....	72
4. PROBLEM ANALYSIS	73
4.1. State-of-the-Practice Investigation in RE.....	73
4.1.1. Research Method.....	73
4.1.2. Background	73
4.1.3. General Level of Practices and Knowledge	74
4.1.4. Requirements Engineering Practices	75
4.1.5. Development Needs and Attitudes to RE Improvement	77
4.1.6. Expectations for Academia.....	78
4.1.7. Discussion and Conclusion of the Study	78
4.2. Organizational Context for the Study.....	79

4.2.1.	<i>SMEs and Small Projects with Low Maturity</i>	79
4.2.2.	<i>Lack of Resources</i>	81
4.2.3.	<i>Uncertainty about Payback</i>	83
4.3.	Application Domain for the Study.....	86
4.4.	Summary	87
5.	SOLUTION CONCEPT FORMULATION	88
5.1.	From the Problem to a Solution.....	88
5.2.	Characteristics of the Solution.....	90
5.2.1.	<i>Specificity</i>	90
5.2.2.	<i>Innovation Characteristics</i>	91
5.2.3.	<i>Other Characteristics</i>	92
5.2.4.	<i>Discussion</i>	96
5.3.	Summary	97
6.	THE BARE METHOD	98
6.1.	Method Components and Relationships	98
6.2.	Method Overview.....	102
6.2.1.	<i>Key Elements in Practice</i>	102
6.2.2.	<i>Requirements Document Template</i>	105
6.2.3.	<i>Requirements Development Process</i>	107
6.2.4.	<i>Method Documentation</i>	109
6.2.5.	<i>Use of the Method</i>	111
6.3.	Limitations of the Method	112
6.4.	Alternatives and Extensions to the BaRE Method.....	113
6.5.	Summary	117
7.	LITERATURE BASED EVALUATION	118
7.1.	REAIMS Top Ten Assessment	118
7.2.	Key Issues in RE Tool and Technique Selection.....	118
7.3.	Software Project Success Factors	120
7.4.	Software Project Risk Factors	121
7.5.	Evaluation against other Requirements Engineering Methods.....	121
7.6.	Discussion and Conclusions	126
8.	EMPIRICAL EVALUATION	128
8.1.	Introduction	128
8.1.1.	<i>Research Methods</i>	128
8.1.2.	<i>Used Assessment Tools</i>	128
8.1.3.	<i>Case Study Overview</i>	131
8.2.	Case Studies	131
8.2.1.	<i>Company Backgrounds</i>	132
8.2.2.	<i>Improvement Actions</i>	134
8.2.3.	<i>Changes in the RE Practices</i>	137
8.2.4.	<i>Achievements</i>	143
8.2.5.	<i>Costs</i>	145
8.3.	Contexts for Change	146
8.3.1.	<i>Evaluation Framework</i>	146
8.3.2.	<i>Environmental Context</i>	149

8.3.3. <i>Organizational Context</i>	151
8.3.4. <i>Technological Innovation</i>	154
8.3.5. <i>Factors Affecting the Case Study Outcomes</i>	172
8.4. Discussion and Conclusion.....	174
8.5. Summary	178
9. CONCLUSIONS	180
9.1. BaRE – an Easy to Adopt Method for Requirements Engineering.....	180
9.2. Guidelines for Easy to Adopt Method Development.....	184
9.3. Future Research	186
EPILOGUE.....	188
REFERENCES.....	189
APPENDICES	
Appendix 1. Requirements Document Topics Survey	
Appendix 2. Case Study Descriptions	
Appendix 3. Competence Evaluation Form	
Appendix 4. BaRE Requirements Document Template	
Appendix 5. BaRE Requirements Development Process	
Appendix 6. RE Practice and Technique Summary	
Appendix 7. Requirements Changes in the BaRE version 1.0	
Appendix 8. Change Requests for the BaRE version 1.0	

1. INTRODUCTION

1.1. Basic Question of the Thesis

Requirements engineering (RE) is one of the hardest tasks in software development. On the basis of studies in hundreds of organizations around the world it has been found that the RE practices are deficient in more than 75% of all enterprises (Jones 1996, p. 228). Furthermore, there are studies reporting the misunderstanding of requirements as the third most important risk factor for software projects (Keil et al. 1998, p. 82) and incomplete requirements as the biggest reason for software project failures (The Standish Group 1994). The problem has also been described in plain words as follows (Brooks 1987, p. 17):

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

Even though the situation in industry leaves a lot of room for improvement, research in the field is not having a break (Nuseibeh and Easterbrook 2000; van Lamsweerde 2000b). On the practical side, the latter half of the 90s saw the emergence of numerous improvements in the field, as standards (e.g. IEEE/EIA 12207.0 1996; ANSI/IEEE Standard 830 1998), generic processes (e.g. Jacobson et al. 1998; Leffingwell and Widrig 1999; Robertson and Robertson 1999), good practice guides (e.g. Sommerville and Sawyer 1997; Wiegers 1999), and comprehensive methods like VORD were published (Kotonya and Sommerville 1998). However, the wealth of literature did not seem to affect the situation in the industry, and recent state-of-the-practice studies still report immature practices (e.g., Kamsties et al. 1998; Nikula et al. 2000b; Juristo et al. 2002). Thus it was concluded that a fundamentally different kind of approach had to be suggested if the industry was to exhibit a different kind of response to the research efforts.

To tackle the problem of deficient RE practices it seemed evident that a proper but easy to adopt RE approach was needed. Developing a proper RE approach as such did not seem like a problem since a wealth of literature already existed. However, the reported body of knowledge appeared so extensive that deciding what to include and what to leave out of the new approach raised uncertainty. To be able to deal with this problem it was decided to start building the new approach based on documenting requirements. The idea of documenting requirements is not a novel one since Gause and Weinberg (1989, p. xvi) state that “The document is nothing; the documenting is everything.”; Sommerville and Sawyer (1997, p. 44) claim it is one of first things an organization should do when improving RE practices; numerous requirements document templates have been suggested (e.g. Kovitz 1998; Leffingwell and Widrig 1999; Wiegers 1999; Pressman 2001a; Robertson and Robertson 2001); and more generally it has been claimed that practice is mediated by artifacts (Suchman 1989, p. 4). Thus it seemed justified to base the new RE approach on documenting requirements and supporting this with systematic practices as appropriate.

The ease of adoption was approached from three different viewpoints. First, diffusion of innovation studies have investigated factors affecting the adoption, and especially the

characteristics of innovations studies (e.g. Rogers 1995) appeared very relevant for the present study. Second, since the local industry could in general be characterized by small size and introduction level practices (Nikula et al. 2000a), so-called lightweight approaches (e.g. Beck 1999b; AgileAlliance 2002) seemed to fit the needs of the local industry better than so-called plan driven methods (cf. Boehm 2002). Third, the role of specificity on the ease of adoption was considered. Newell (1969) established the notion of *strong* and *weak* problem solving approaches in which strong methods are designed to solve only specific problems, while weak methods can be used to solve many types of problems. Another important aspect of the strong methods is that they provide detailed help in solving the problems they are designed to address, while weak methods only provide superficial help to the large group of problems they address. These ideas were the basis for the Vessey (1991) study that proposed the notion of cognitive fit which means that “complexity in the task environment will be effectively reduced when the problem-solving aids (tools, techniques, and/or problem representations) support the task strategies (methods and processes) required to perform that task” (p. 220). Vessey claims further that cognitive fit results in increased problem solving efficiency and effectiveness. Thus it was hypothesized that by limiting the applicability of the suggested solution to a specific domain, it would be possible to increase the fit between the problem domain and the solution and, thereby, ease the adoption of the solution. Again based on the needs of the local industry, it was decided to focus on the small administrative and business applications domain (Association for Computing Machinery 1998). In short, the goal of the present study became to develop a lightweight but document-driven RE method that would be easy to adopt by small organizations having initial level RE practices and developing small administrative and business applications.

The identification of the central characteristics for the desired method made it possible to study whether such methods already exist. For example, the Rational Unified Process (RUP) gave an easy to adopt impression, but a closer look at the Roadmap to Small Projects (Rational Software Corporation 2002) outlined the steps to modify the full-blown RUP to a small project, suggesting that using RUP without modifications in a small project was not feasible. Some agile methods, on the other hand, had detailed instructions on how to do actual development (e.g. Beck 1999b; Cockburn 2002), but their support for RE was limited. In particular documenting requirements was rare which is well demonstrated by Cockburn (2002, p. 175): “Ideally, documentation activities are deferred as long as possible and then made as small as possible.” Overall it appeared evident that no comparable lightweight but document-driven RE method had been suggested before, and the basic question of this thesis developed into whether domain specificity would make it possible to develop a method that would help the industry improve RE practices.

1.2. Research Questions

The fundamental goal for this research effort was to support daily software development in the area of requirements engineering. The study was done in two phases, the first of which was an exploratory phase and concentrated on figuring out what the software industry needed the most in the RE area. This phase consisted of three steps that clarified the state of the practice in RE in small and medium enterprises (SMEs), sought reasons for immature practices found, and closed with the construction of a basic RE method – the BaRE method. The second phase focused on the evaluation of this method and studied how it worked in practice. Thus the research question for the present study can be summarized as *how to ease the adoption of basic systematic RE*

practices in small organizations. This question is further divided into four more specific questions:

1. What are the desirable characteristics of a requirements engineering method that would make it easy for small organizations to adopt it?
2. Is it possible to develop an easy to adopt method for requirements engineering?
3. What are the desirable characteristics of the introduction process of an easy to adopt requirements engineering method in small organizations?
4. How to develop easy to adopt methods?

The first question is addressed by conducting a problem domain analysis and developing a solution concept that is expected to address the problems found. The second question is tackled by constructing a new requirements engineering method. The third question is addressed by studying the experiences from the adoption of the constructed method in three small organizations and the last question is answered by summarizing the experiences from the method construction and introduction activities.

1.3. Research Methods

The predominant research method in the present study is constructive research which is supplemented by literature surveys, a state-of-the-practice investigation in industry, and case studies. The reason for selecting the constructive approach was an interest in doing practically relevant research and a constructive approach was reported to suit that task well (Lukka 2002). Section 1.3.1 presents different views on constructive research and the way they are addressed in the present study. Section 1.3.2 describes the key issues in case study research from the point of view of the present study.

1.3.1. Constructive Research

Constructive research builds on both prior theory and the practical relevance of the work (Figure 1). In the present study, *connection to prior theory* refers to the research results from previous studies (Chapter 3) while the *practical relevance of the problem and the solution* refers to the conducted problem analysis (Chapter 4) and solution concept development (Chapter 5).

The central element in a constructive research is the implementation of the *solution to the initial problem* (Chapter 6) and the *practical functioning of the solution* was confirmed with an empirical evaluation in three companies (Chapter 8). The *theoretical contribution of the study* includes the development of an easy to adopt solution concept to solve the identified problems and the identification of its desirable characteristics (Chapter 5), insights about process improvement in small organizations (Section 8.4) and a proposal for general guidelines for easy to adopt method development (Section 9.2).

A typical constructive research process is presented in Figure 2. The emergence of this thesis is described in more detail in Section 1.5, but the main phases are problem identification, refinement, solution search, method construction, and evaluation, which match Figure 2 in the following way: a *practically relevant problem* was identified in the problem identification and refinement phases (Section 3.1 and Chapter 4), keeping in mind the *potential for research co-*

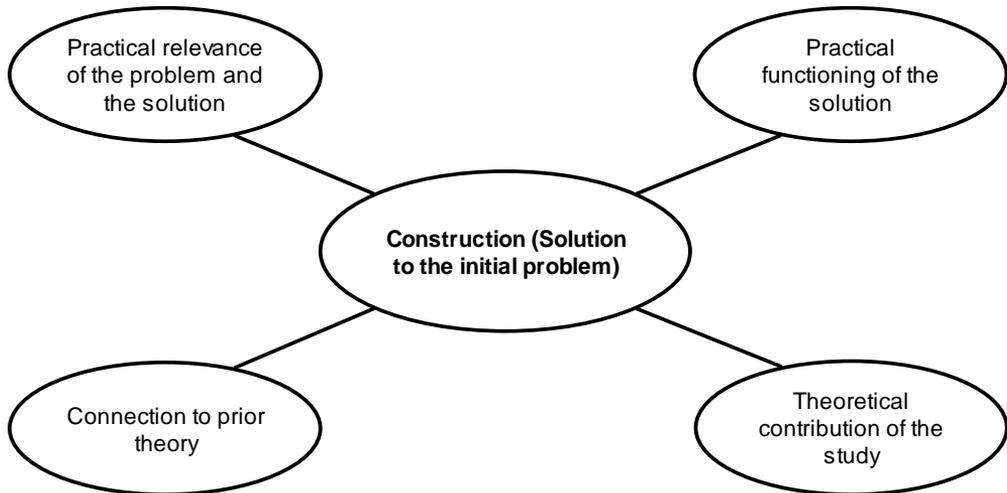


Figure 1. The central elements of the constructive research approach (Lukka 2002)

operation throughout the phases (Section 1.5). A *deep understanding* of the domains related to the effort was acquired in the problem refinement (Chapter 3), problem analysis (Chapter 4), solution concept development (Chapter 5), and construction phases (Chapter 6). The *innovative part of the work* was done mainly in the problem analysis (Chapter 4) and solution concept development phases (Chapter 5), but the contemplation and refinement of ideas continued also in the *implementation phase* (Chapter 6). *Testing the solution* was done in the evaluation phase first as literature based evaluation (Chapter 7) and then as empirical evaluation in industry (Chapter 8). *The scope of the applicability* and *the theoretical contribution* of the study are summarized in Chapter 9.

Also the articles by Nunamaker et al. (1990) and March and Smith (1995) discuss constructive research even though both have chosen to name their research approaches differently. Nunamaker et al. talk about system development as a research methodology and specifically

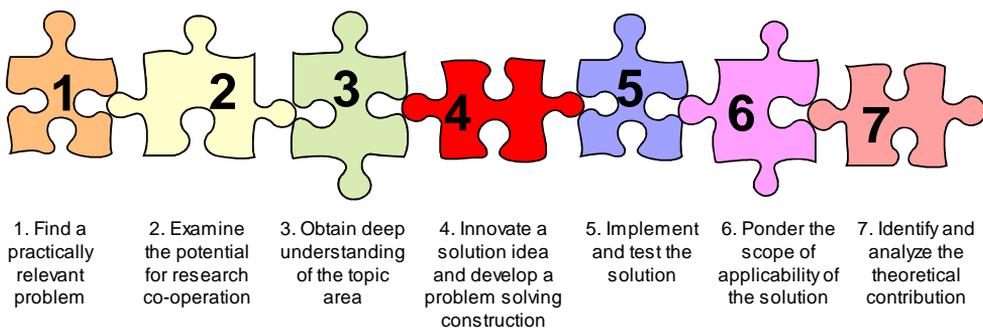


Figure 2. A typical constructive research process (Lukka 2002)

about *developmental research* while March and Smith explain how both natural science and *design science* are needed in information technology research. According to Nunamaker et al., the idea of system development as a research methodology fits comfortably into the applied science research category and further belongs to the engineering, developmental, and formulative types of research. The appropriateness of systems development as a research method is not self-evident and this has been questioned by others (Järvinen 1999, p. 60). However, considering the Nunamaker et al. article from a more general constructive research point of view, it provides a useful framework for the present study and the following two observations become relevant:

- “[the research artifact/concept] at issue is likely to be viewed for its applications value rather than for its intrinsic value. This suggests that a concept with wide-ranging applicability will go through a research life cycle of the form: concept–development–impact.” (Nunamaker et al. 1990, p. 91)
- “...the developed system serves both as a proof-of-concept for the fundamental research and provides an artifact that becomes the focus of expanded and continued research. Contributions at each stage of the life cycle obviously contribute to *fuller scientific knowledge of the subject*.” (Nunamaker et al. 1990, p. 92)

Nunamaker et al. suggest that a multimethodological approach to information systems research consists of four research strategies: theory building, experimentation, observation, and systems development. The theory building includes method development which is a central part of the present study and covers also the method documentation (Nikula 2002b). The present study includes experimentation that has been published elsewhere (Mannio and Nikula 2001; Reinikainen et al. 2001), and the observation part of the study is covered in Section 4.1 and Appendix 2 (Case Study Descriptions). In the present study, no system was developed in the Nunamaker et al. sense but method prototyping and technology transfer aspects of this research strategy were implemented with a workshop with industry participants to get early feedback on the method, and case studies were used to evaluate the acceptance of the proposed method.

The central tenet in the paper by March and Smith (1995) is that information technology (IT) research should be built on both natural science and design science. The goal of the natural science part is increasing the understanding of the nature of IT while the design science aims at improving IT performance. To achieve these goals the *natural science* part of IT research tries to discover (or theorize) and justify scientific claims. In the same vein the *design science* consists of *build and evaluate* activities from which the build part tries to construct things that serve human purposes and *evaluate* part assesses their value or utility. Thus the March and Smith view on IT research produces the following outline for the present study:

1. Conduct a problem domain analysis to understand the problems (Chapter 4) and develop a solution concept to address the identified problems (Chapter 5). That is, it is first theorized what kind of solution should be built to address the identified problems.
2. Build an artifact to solve the problems, show that a solution can be constructed (Chapter 6).
3. Evaluate the built solution (Chapters 7 and 8).
4. Theorize the evaluation results, how and why the method worked in practice (Chapters 8 and 9).

In the present study, the findings of the evaluation are reported and reasons for the outcomes are discussed but no theory as such is developed due to the small scale and exploratory nature of the study. March and Smith provide only superficial advice on the justification work (1995, p. 262): “The *justify* activity performs empirical and/or theoretical research to test the theories posed.” In the present study these two natural science activities are left as topics for further study.

1.3.2. Case Study Research

The method evaluation phase of the present study followed the case study approach as described by Yin (2002). The quality of the case study research can be judged by four tests shown in Table 1 (Kitchenham et al. 1995, p. 55; Yin 2002, p. 33): construct validity, internal validity, external validity, and reliability. Even though Kitchenham et al. focus in their article on method evaluation with case studies, the Yin approach was followed since it provides a more comprehensive discussion on case studies.

Table 1. Tests for the quality of empirical social research (Yin 2002, p. 34)

Test	Purpose
Construct validity	Establishing correct operational measures for the concepts being studied
Internal validity (for explanatory or causal studies only, and not for descriptive or exploratory studies)	Establishing a causal relationship, whereby certain conditions are shown to lead to other conditions, as distinguished from spurious relationships
External validity	Establishing the domain to which a study’s findings can be generalized
Reliability	Demonstrating that the operations of the study – such as the data collection procedures – can be repeated, with the same results

In the present evaluation project the four criteria for research design quality were addressed in the following ways. To increase the *construct validity*, multiple sources of evidence were used: all the developed requirements documents were collected from the projects completed in the case studies and other related documents when possible (e.g., submitted change requests, developed change management guides and document templates, and review records). In some cases it was also possible to obtain some archive records showing the planned and completed activities with time statistics. The conducted interviews included survey, focused, and open ended questions to collect data for specific questions. To establish a chain of evidence, the meetings during the study were audio recorded, transcribed, and analyzed with the ATLAS.ti application for qualitative analysis (Scientific Software Development 2004). A separate diary was kept for each case study where all the contacts were recorded in brief and all the emails and documents were stored for further study in the analysis phase. The case study descriptions are included as Appendix 2 in this thesis and the key issues from the point of view of this study are reported in Chapter 8. The case study descriptions (Appendix 2) were reviewed and accepted by the key informants participating in the case studies.

The conducted case studies are descriptive in nature but they also include explanatory elements, since the answers to the research questions had to be concluded from observed and measured events. Thus the *internal validity* of the results was important and it was addressed mainly by pattern-matching and some explanation building in the data analysis phase. The *external validity*, on the other hand, was addressed already in the research design phase by using a multiple-case study design with replication logic. Finally, the *reliability* was addressed by using the same protocol for all three cases and developing a data base from the collected data. The case study protocol proved important also from the project management point of view. Namely, having three parallel case studies made it necessary to handle all the cases with one planned protocol, and the fact that the case studies were run physically at a distance from the researcher forced basically all the meetings to be arranged within two to three day time frames. Consequently, all the cases got very similar treatments during the study.

A common issue with case study research is the difficulty of generalizing the results (Yin 2002, p. 37). For example Kitchenham et al. (1995, p. 55) state that “For experimental results to be generalized, there must be either two alternative treatments or a well-defined control.” In the industrial setting of this study both of these options were clearly infeasible. Thus the generalization of the results of the present study relies on analytical generalization in the same way as in case studies in general (Yin 2002, p. 36).

1.4. Research Contribution

The key contribution of the present study is the development and validation of an easy to adopt method for RE. A problem domain analysis identified two typical problems in the studied organizational setting (Chapter 4), and to be able to resolve these problems an easy to adopt solution concept with thirteen desirable characteristics was developed (Chapter 5). The constructed BaRE method is described in Chapter 6 and its evaluation is reported in Chapters 7 and 8.

The evaluation project resulted in a number of various insights about process improvement in practice (Chapter 8). First, the project provided evidence of the benefits of domain specific software engineering in the RE area. The adaptability of the BaRE method made it possible to use both evolutionary and revolutionary approaches in process improvement, and both of them proved to have distinctive benefits. All the adoption projects themselves were different and provided further understanding of the adoption processes in general. Finally, the BaRE method evaluation project resulted in the improvement of RE practices in the participating companies providing evidence that RE practice improvement does not necessarily require extensive resources and time scales. The experiences from the conducted study are summarized and reflected upon in Section 9.2 as issues to consider when developing easy to adopt methods.

The development and evaluation of the method required the development of supporting infrastructure. Consequently, a requirements document template (Appendix 4), summaries of requirements development and management practices, and training material for the method were developed (Nikula 2002b). The method evaluation, on the other hand, required metrics to assess the current practices and changes in them. To be able to do that, an existing assessment framework was adapted to the domain of interest and a Lightweight REAIMS Top Ten assessment and an infrastructure checklist were developed to measure changes in the RE practices (Sections 3.2.1, 7.1, 8.1.2, and 8.2.3).

1.5. Emergence of the Thesis

This section describes the main events in the course of this study. For a reader who is only interested in the outcome of the work, this section is of marginal interest. However, in some cases the outcomes may be hard to understand unless the origin is known, and this section is included to provide that. This section provides also the history for the BaRE method described in this thesis as suggested by Introna and Whitley (1997, p. 242). The events are described in chronological order starting from *problem identification*, *refinement*, *solution search*, *method construction*, *evaluation*, and closing with the *manuscript preview process*.

Problem Identification. I encountered problems in practical RE as a part of daily work in software industry. It is no surprise that things were not easy for a novice, but the fact that even older colleagues felt the same way was disturbing. Luckily, a professor had a book on the topic (Sommerville and Sawyer 1997) that shed some light into the darkness. Studying other related literature (Humphrey 1989; Davis 1993; Brooks 1995; Davis 1995; Cusumano and Selby 1996; Haikala and Märijärvi 1997; ANSI/IEEE Standard 830 1998) started to increase my basic knowledge in the field, but at the same time numerous questions arose. An initial analysis of the situation resulted in the view that research had already produced a number of solutions to practical RE problems, but for some reason they just had not been transferred to – or adopted by – industry. Consequently, the conclusion was that RE research should produce something that supports the daily software development in a concrete way from the software developer's point of view; the practical idea was to develop a requirements document template suitable for small projects and provide tool support for it. A stereotyped notion that universal solutions are not feasible led to an initial decision to focus on SMEs.

Problem Refinement. Starting work as a researcher made it possible to study the initially identified problem closer and refine it into an academic research effort. The study started with RE conference proceedings (ICRE 1996; RE 1997; ICRE 1998), which provided an overview of typical research efforts in the area. Other efforts to increase the understanding of the prevailing situation included studying basic books in RE (Gause and Weinberg 1989; Jackson 1995; Sommerville and Sawyer 1997; Thayer and Dorfman 1997; Kovitz 1998; Leffingwell and Widrig 1999; Robertson and Robertson 1999), the state of practical surveys in RE (Curtis et al. 1988; Lubars et al. 1993; El Emam and Madhavji 1995; Kamsties et al. 1998; Weidenhaupt et al. 1998), literature on RM tools (Tvette 1999; INCOSE 2003), technology transfer literature (Rogers 1995; Siddiqi and Shekaran 1996; Harel 1997; Miller 1997; Fowler et al. 1998; Morris et al. 1998; Regnell et al. 1998; Moore 1999; Software Engineering Institute 2002), and more general literature on software engineering and research (STRÍ TS2 1991; Galliers 1992; Fenton et al. 1994; Pfleeger 1994; Kitchenham 1996; Pfleeger 1998; Pressman 1998; Sommerville 1998; Zekowitz and Wallace 1998). The familiarization in the topic included also participating in an RE conference (RE 1999), workshops, and seminars. Since practical relevance was one of the corner stones for the work, a state-of-the-practice study was conducted in 12 companies (Nikula et al. 2000a; Nikula et al. 2000b; Nikula et al. 2000c). One section of this study concerned interest in participating in a project to develop RE practices, but did not lead to concrete actions due to lack of time caused by other duties at the university.

Both the literature studies and the state-of-the-practice study confirmed the original observation that RE practices were indeed immature in the industry and few companies were doing it properly. An analysis of the situation led me to believe that in addition to a proper requirements

document (RD) template and tool support, also systematic practices, RE processes, RE metamodels, metrics, prototypes, training, requirements representation and specification issues had to be addressed. The domain specific characteristics were refined to introduce and improve RE practices in small software projects in SMEs.

Solution Search. Having been assured that the original problem had not been solved, the development of a new solution was started. The target application domain was refined to include administrative and business applications (cf. Association for Computing Machinery 1998) in addition to small software projects with immature RE practices, since the domain specific approach seemed the only feasible way to continue (Glass and Vessey 1998). Due to the focus on SMEs it was concluded that a lightweight approach would suit the target companies best (Beck 1999a; Beck 1999b). Finally, it was decided that the solution should be based on systematic practices covering the whole RE area in a comprehensive manner, and everything should be bundled in a single package. The last set of decisions was based on personal preferences and intuition. Having an idea of the solution characteristics experimenting with the ideas was started and positive feedback from an industrial partner was received (Mannio and Nikula 2001; Reinikainen et al. 2001).

Developing a proper solution was not possible with the shallow knowledge in the fields of interest and research in general, so a closer look at various topics was required. First the differences between systems, software systems, and software engineering were studied (Carter et al. 1988; McAuley 1992; Harbaugh 1993; Harwell et al. 1993; Downward 1994; Oliver 1995; Forsberg and Mooz 1997; Thayer 1997; Flynn 1998); knowledge of the requirements management tools was increased (James 1996; LaBudde 1997; Meilich 1997); and RE in general was studied (Scharer 1981; Goguen and Linde 1993; Maiden and Rugg 1996; Nuseibeh and Easterbrook 2000; van Lamsweerde 2000b). A study of traditional systems engineering in the form of control and feedback systems (Söderström and Stoica 1989; Blanchard and Fabrycky 1990) appeared difficult to apply in the selected domain and it was excluded from further study. However, a book called “Rethinking Systems Analysis & Design” (Weinberg 1988) proved thought provoking and raised the issue of rethinking RE, which resulted in an observation that the observed research efforts generally focused on complex problems with detailed solutions. That is, basic level RE targeted for companies with immature practices was quite limited (Fowler et al. 1998; Tvet 1999) even though technology transfer issues remained active (Kaindl 2000; Greenspan 2001; Kaindl et al. 2002). At the same time the literature on method engineering provided further ideas on method construction from method elements (ter Hofstede and Verhoef 1997), terminology (Brinkkemper 1996), and on reasons why method engineering (ME) as such appeared to fail (Introna and Whitley 1997). Among the most important results of the ME study was renaming the former “package” a method (Odell 1996) and continuing its development in the ready-to-hand direction (Introna and Whitley 1997). Settling in a method construction task resulted also in advances in the research arena since it fixed the research method to a constructive one (Galliers 1992; March and Smith 1995; Järvinen 1999). An initial idea for the method was to develop it on the basis of problem frames (Jackson 1995; Jackson 2001), but a closer study of the concept revealed it so different from traditional approaches that the effort required to learn it appeared unrealistic. Thus the decision to focus on basic good RE practices (Sommerville and Sawyer 1997; Wiegers 1999) and a common sense approach to requirements documents was found feasible (Kovitz 1998).

Towards the end of this phase the anticipated method elements did not change except for finding feasibility study, requirements modeling, and project management also important. At

this point the method was tentatively called a minimum RE method as it became clear that help was needed most in introducing RE practices into the industry. With these ideas a negotiation round in local SMEs was done to explore their interest in cooperation. However, suitable SMEs were few in number and even their interest in RE did not suffice for investing the time and money the proposed action research would have required. Consequently, a decision was made to construct the method on the basis of existing knowledge and the role of industry was left to evaluating the completed method.

Method Construction. Having established a reasonable working set of method requirements and components, the actual construction task became possible. To make it easier to discuss the method, it was given a name derived from the words minimum RE – MiRE. The first actual development task was reviewing standards on RD related topics (STRÍ TS2 1991; e.g., ISO/IEC 12119 1994; IEEE/EIA 12207.1 1997; ANSI/IEEE Standard 830 1998; ANSI/IEEE Standard 1362 1998; ISO/IEC 9126-1 2001) and developing an RD template based on them and other published templates (Kovitz 1998; Leffingwell and Widrig 1999; Wieggers 1999; Pressman 2001a; Robertson and Robertson 2001). The rest of the method – requirements development and management practices, tool support for RM, and training – were adapted from RE good practice guides and other RE books (Sommerville and Sawyer 1997; Kotonya and Sommerville 1998; Robertson and Robertson 1999; Lauesen 2002). A literature survey was conducted based on the initial ideas of the method to find comparable ones, but none of the found methods could be considered a real rival for the planned method.

During the construction phase the lightweight property started getting an increasingly important role, and finally a closer study on the subject was in place. This study revealed an increasing body of literature on the topic with an established term *agile methods* (AgileAlliance 2002). The literature on the topic proved invaluable for the present study since in addition to providing support for the earlier observations also a categorization of expertise levels (Cockburn 2002) and a comparison between agile and plan driven methods (Boehm 2002) were found.

Other events in the construction phase included a lesson in the English language pointing out the dictionary meaning of the word “mire” which led us to change the name to BaSyRE for **basic systematic RE**. The method was introduced with this name in a workshop (Nikula and Sajaniemi 2002), which raised the issue of readiness-to-hand. Namely, the concept appeared new and hard to understand both to the paper referees and the workshop participants. Using a non-intuitive concept was contradictory to the principle goals for the method, and a quick fix was made by replacing only the phrase *ready-to-hand* by *ready-to-use*. Further refinement of the idea resulted in abandoning the whole ready-to-hand idea as defined by Heidegger (Heidegger 1927; Winograd and Flores 1986; Dreyfus 1991) and adopting the common sense meaning of the phrase ready-to-use to provide a receiver-oriented name “so that the word symbol for a new idea has the desired meaning for the intended audience” (Rogers 1995, p. 237). This approach is consistent with Riddle and Fairley (1980, p. 1) who classify methods as cognitive tools; this classification means also that methods cannot be ready-to-hand in the Heidegger sense. Another observation from wider discussion on the method was that the name was too complicated; thus it was decided to keep it simple and the name was revised to BaRE for **Basic RE**, pronounced [ˈbeə(r)].

On the industry front the first official seminar in tSoft, a software engineering technology transfer program at the University of Joensuu, Finland (tSoft 2002), was held soon after the beginning of this phase, and the final recruiting of companies to test the method in practice was

started. In addition to this general seminar, a workshop dedicated to the BaRE method was held where companies got a closer look at the method and its design. In the negotiation phase four companies volunteered to use the method in their daily work but by the time the method was completed, one company had to step back from the project for lack of suitable projects.

Evaluation. The method construction phase was closed with the publication of the BaRE method version 1.0 (Nikula 2002b), but the research continued with method evaluation. Desktop evaluations of the method by the licentiate thesis reviewers and industry representatives adopting the method suggested that the method attained the goals of being practical and simple. The principles of the BaRE method seemed compatible with the general interests in the field, since work with comparable ideas had been undertaken by other people (e.g., Breitling et al. 2002; Davis 2002b; Durán et al. 2002; Hardiman 2002; Kauppinen et al. 2002; McPhee and Eberlein 2002; Nawrocki et al. 2002; SOPHIST Group 2002; Vickers et al. 2002). The only major change in the foundations of the method after its publication was revision of the method characteristics. A number of changes were suggested by the end of the method evaluation phase, and they are documented in Appendix 7 (Requirements Changes in the BaRE version 1.0). The empirical evaluation was conducted as planned, and is documented in Chapter 8 and Appendix 2 (Case Study Descriptions).

Manuscript preview process. The dissertation manuscript was a subject of many improvement ideas in the course of the preview process. The major changes implemented during the preview phase included a move to a more systematic and accurate reporting of the study based on published research methods; strengthening the theoretical basis of the study by increasing the number of references by over 50% and also by moving the focus to more scientific literature; and finally the readability of the thesis was improved by numerous refinements and some major structural changes in presentation. Overall, a fairly technology transfer focused reporting of the study was refined to a more adopter aware approach which is also reflected by the differences between the title of this doctoral thesis and the licentiate thesis used as the basis for the present work (Nikula 2002a).

1.6. Structure of the Thesis

The rest of this thesis is structured as follows. Chapter 2 defines the basic RE terms used in the present work and Chapter 3 takes a look at literature on topics related to this thesis. Chapter 4 provides a problem analysis of the perceived problem and Chapter 5 explains the solution concept development. Chapter 6 describes the constructed BaRE method itself. A literature based evaluation of the method is provided in Chapter 7, and empirical evaluation is documented in Chapter 8. The thesis is closed with conclusions in Chapter 9.

2. TERMS AND DEFINITIONS

This chapter introduces the basic terms and definitions used in this thesis. The RE terminology is still in a state of flux and there seems to be little chance of establishing a unanimously accepted terminology. Thus only basic concepts and terms are introduced, and competing terms are explained when there is a clear risk of confusion. The definitions are based on literature targeted for people unfamiliar with RE and are intended to provide a clear and systematic view on RE even though this may neglect some important practical aspects of software development.

This chapter consists of four subsections. First Section 2.1 discusses the context of this thesis and Section 2.2 introduces the key terms in the RE area. Section 2.3 defines the terms used for requirements documentation and Section 2.4 summarizes the terms related with methods, their adoption, and use in the present study.

2.1. Context

Requirements engineering is a part of software, systems, and information systems development work. Even if the RE-related activities are fairly well agreed upon, the term RE is not recognized so well, and competing terms like requirements management (RM) (Hooks and Farry 2000), requirements analysis (Jalote 1991), and requirements determination and analysis (Flynn 1998) are often used. Thayer and Dorfman (1997, p. 510) provide a good starting point by defining *requirements engineering* as follows: “In systems engineering, the science and discipline concerned with analyzing and documenting requirements. It comprises needs analysis, requirements analysis, and requirements specifications.”

Since software, systems, and information systems fields were separated above, the differences between the fields are outlined next. *Software* can be defined simply as “computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system” (IEEE Standard 610.12 1990, p. 66) and *software engineering* as “(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1)” (IEEE Standard 610.12 1990, p. 67). The term *system*, on the other hand, has a much wider range of definitions and two competing definitions are considered here that help to understand the scope of the topic better. A simple definition provides a good starting point: “A system is a purposeful collection of interrelated components that work together to achieve some objective” (Sommerville 2001, p. 21). However, as this is quite a general definition, a more complete one is also in place (Thayer and Dorfman 1997, p. 518):

System. (1) A collection of hardware, software, people, facilities, and procedures organized to accomplish a common objective. (2) A set of components that interact according to a design. A component of a system can also be another system (called a subsystem). Such components (subsystems) may be: controlling or controlled system; hardware, software, human interaction; input and output subsystems. (3) A bounded physical entity that achieves within its domain a defined objective through interaction of its parts.

An *information system* differs from a system by two key aspects. First it takes into account the organizational context of a system which is also reflected in many definitions, e.g. Flynn (1998,

p. 3): “An information system provides procedures to record and make available information, concerning part of an organization, to assist organizational activities.” Flynn continues to divide his definition into two parts of which the first part emphasizes system structure and functioning and the second part is concerned with the organizational context of the system. Another important aspect of information systems is that even if the technical implementation of systems from hardware components is recognized, they are usually taken as given and, e.g., information systems research does not normally include the design and implementation of dedicated hardware.

The context of this work includes two more terms that call for definitions, namely *application domain* and *small and medium enterprises*. The selected *application domain* for this work is administrative and business applications which is taken from the Association for Computing Machinery (1998) computing classification system. The reason to focus on a limited application domain is to balance between the applicability and the fit for the need of the suggested solution. That is, generality is often inversely related with the provided support for any specific task and thus seeking for a ready-to-use solution was considered a realistic goal only in a limited domain. An example of properties that are normally not addressed in the selected domain is time constraints which, on the other hand, are inherent in embedded systems.

The present work is expected to be most applicable in *small and medium enterprises* (SMEs) since small projects are often a natural choice for them. The European Community definition for small and medium enterprises limits the number of employees to less than 250 (The European Commission 2003) while the other requirements for an SME state that the annual turnover must not exceed 50 million euros and/or an annual balance sheet total must not exceed 43 million euros. Furthermore, less than 25% of the enterprise capital or voting rights may be owned or controlled by non-SMEs.

2.2. Requirements Engineering

The requirements engineering domain can be divided into requirements development and management activities where the *requirements development* covers elicitation, analysis, documenting, and validation tasks (Figure 3). *Requirements elicitation* deals with eliciting (or capturing, trawling, finding, etc.) requirements from different sources in a project. Once the requirements have been collected, *analyzing* them for completeness, conflicts, errors, overlap etc. can be done. The *documenting* task organizes the requirements usually in a form of a requirements document. Requirements documents – and especially the requirements in them – are *validated* by the customer before the development commences. (Notice that the term *verify* has been used by Thayer and Dorfman (1997, p. 1) and Wiegers (1999, p. 19), the term *validate* by Sommerville and Sawyer (1997, p. 189) and Kotonya and Sommerville (1998, p. 87), and *tested in quality gateway* by Robertson and Robertson (1999, p. 181); here the term *validate* is used for this activity (Boehm 1984, p. 75).) Once the document is ready and validated, the *requirements management* can start. It is responsible for “establishing and maintaining an agreement with the customer on the requirements for the software project” (Software Engineering Institute 1995, p. 126).

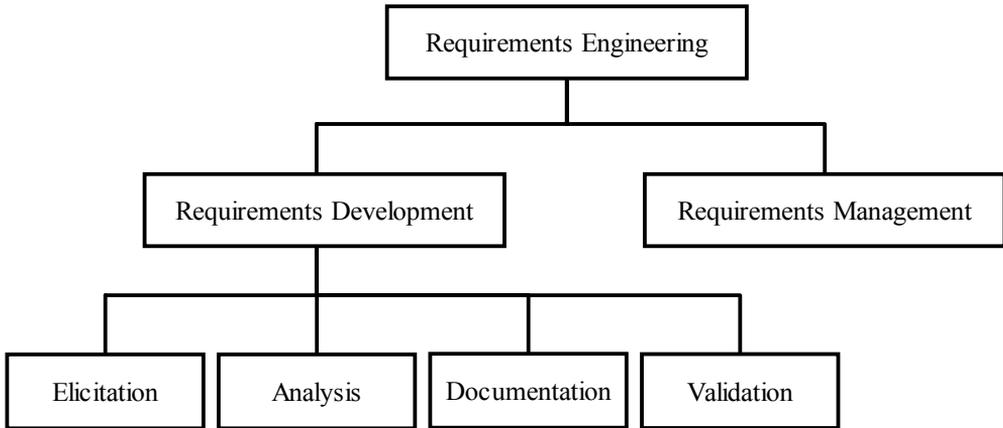


Figure 3. Hierarchical decomposition of the requirements engineering domain (adapted from Wiegers 1999, p. 19)

Figure 4 shows the interplay and the boundary between requirements development and management. The requirements development phase focuses on developing baselined requirements before actual software development, and once the development is started, the requirements are managed through a Requirements Change Process.

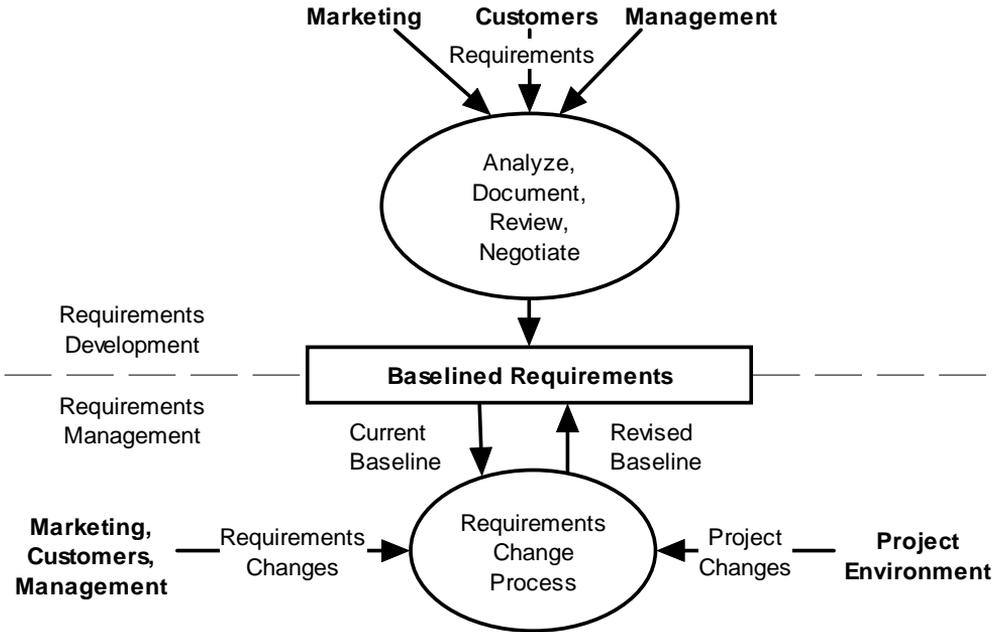


Figure 4. The boundary between requirements development and management (Wiegers 1999, p. 21)

Change management and *requirements management* are related but separate activities. The principle difference between these activities is that change management covers all kinds of change requests (e.g., problem reports, change requests, new requirements, and enhancement requests) while requirements management focuses on requirements and addresses all issues related to their handling (e.g., identification, storage, retrieval, version control, relation to other requirements, change requests, and new requirements). Overall change management is a general activity that can be used to embrace all the change-related activities but requirements management is meaningful only in the context of requirements. If requirements are handled as a *requirements document* without unique identifiers, history, rationale, and alike, it seems quite counterintuitive to talk about requirements management, and possible changes are quite naturally handled as document level *change management*. In practice, *requirements management* becomes a realistic approach when requirements are identified individually and, on the other hand, if requirements are stored in a database based system separately from change requests it is counterintuitive to even address requirements related issues under *change management* after they have been identified as such. In the present study, no clear difference is made between these two terms since in a low maturity organization a gradual transition between these approaches can be expected even though institutionalizing the difference is not likely to happen quickly. However, the rationale presented above can be used to explain the contexts of these terms.

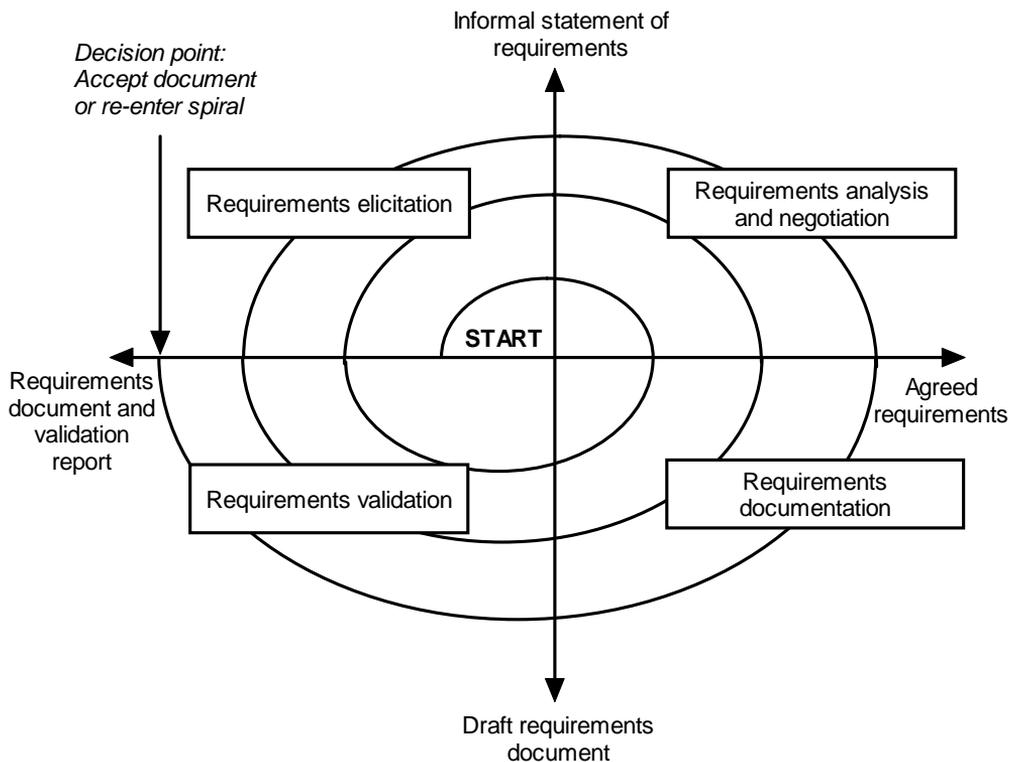


Figure 5. A spiral model of the requirements engineering process (Kotonya and Sommerville 1998, p. 35)

A *requirements engineering process* is usually focused on the requirements development phase. Even though there is no universal requirements process (Sommerville and Sawyer 1997, p. 366), Figure 5 shows a generic spiral model that is in broad terms compatible with most proposed processes (cf. Sommerville and Sawyer 1997, p. 366; Kotonya and Sommerville 1998, p. 35). As the spiral suggests, requirements engineering is iterative rather than linear – the same requirements elicitation, analysis and negotiation, documenting, and validation activities may be repeated multiple times before the process is terminated. The termination of the process is not fixed, since each iteration further refines requirements and should also lead to a higher quality requirements document. The radial arms represent the increasing amount of information generated in each activity and also the increasing total cost of the process.

This Kotonya and Sommerville model provides a very simplistic view of an RE process from the whole software development point of view since it neglects the intertwining of RE and implementation. The intertwining has been described in more detail by various authors (e.g. Swartout and Balzer 1982; Iivari 1987; Jacobson et al. 1998, p. 11; Nuseibeh 2001), and a number of solutions have been proposed for addressing the problem from high level approaches like a spiral model (Boehm 1988) to evolutionary development (Sommerville 2001, p. 9) to refactoring in agile software development (e.g. Beck 1999b, p. 107), and to academic ones (Kerola and Järvinen 1975; Iivari 1983; Iivari 1990). However, in the present study the Kotonya and Sommerville approach is adopted and only the RE phase is discussed here without relating it to the whole software development lifecycle.

In the present work, the *competence* of a practitioner in the RE area is touched upon based on the work reported in Vickers et al. (2002). The topic is not addressed extensively but its importance is acknowledged and explorative work in this area is done. In particular a *competence level* scheme shown in Table 2 is introduced. This scheme is used to indicate the level of competence a person has to perform a task on a five level scale. The conducted competence level evaluation represents rather a practical than a scientific approach to the topic. The information available on the Vickers et al. framework is very limited and, for example, there is no knowledge of the theoretical groundings or validation of the framework. However, published frameworks for competence evaluation are not very common in literature and, for example, the people capability maturity model (Curtis et al. 1997; Curtis et al. 2001) provides little help in assessing employee competence in RE.

Table 2. Five competence levels used in the present work

Level of competence	Characterization
Novice	New to topic in question
Trainee	Knows theory
Supervised practitioner	Can do; supervision and pre-reviews are advantageous
Practitioner	Has done; works independently and participates in peer reviews
Expert	Proven ability; experienced in different domains and adaptation

2.3. Requirements Documentation

In the present study, the basis for requirements documentation is a *Requirements Document* as suggested by Kovitz (1998, p. 130). Two other documents with a comparable focus on customer requirements with customer terminology include the Concept of Operations Document (ANSI/IEEE Standard 1362 1998) and the Requirements Definition Document (Pfleeger 1998, p. 171). There are also a number of document templates for developer oriented documents including the Software Requirements Specification (SRS) (ANSI/IEEE Standard 830 1998), the Requirements Specification Document (Pfleeger 1998, p. 172), and the (Interface) Specification (Kovitz 1998, p. 139). Thayer and Dorfman (1997, p. 515) have suggested the following definition for an SRS:

Software requirements specification (SRS). (1) In software system engineering, a document that clearly and precisely describes each of the essential requirements (functions, performance, design constraints, and quality attributes) of the software and the external interfaces. Each requirement is defined in such a way that its achievement can be objectively verified by a prescribed method, for example, inspection, demonstration, analysis, or test. (2) Data item description of a software requirements specification.

In the present study, the term Requirements Document is used as it seems quite sufficient for the small projects this work focuses on. There is, however, an undeniable need for a clear and precise description of each requirement. Consequently, the above definition of an SRS is kept in mind, and an Interface Specification is included in the document set to have a place for precise developer oriented documentation. It is also important to notice that the term *document* is used in a very general way. An actual requirements management system could be comprised of a data store and different views of it as paper documents or web pages; one of such views could then serve as a Requirements Document.

In addition to the requirements document, also the term *requirement* deserves a proper definition. Kovitz (1998, p. 35) defines requirements quite generally as “the effects that the computer is to exert in the problem domain, by virtue of the computer’s programming.” Another interesting and more precise definition has been suggested for a system requirement (Thayer 1997, p. 93):

System requirement. (1) A system capability needed by a user to solve a problem or achieve an objective, and/or (2) a system capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

The key difference between the given requirement definitions is that the system requirement explicitly states the user need in solving a problem or achieving an objective. The general requirement definition is focused on the software engineering – or programming – viewpoint: what is the computer to do in a given situation. Since software is a subset of a system, the difference between software and system engineering is well reflected by the difference in the term requirement. However, an important point is that a system has an objective which it should accomplish through co-operation with its software, hardware, people, facilities, and procedures. In this thesis, the focus is on administrative and business applications that normally do not

address hardware components, but are closely linked to people and procedures. Thus it is important to consider also people and procedures of the system view.

2.4. Methods

A method construction effort is a central part of the present study and therefore the key terms related to methods, their adoption, and use are introduced next. First of all the definition of a *method* is adopted from Brinkkemper (1996, p. 275): “A method is an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products.” Since term *methodology* is often also used in this context, the Brinkkemper (1996, p. 276) definition of it is in place here: “The methodology of information systems development is the systematic description, explanation and evaluation of all aspects of methodical information systems development.” Brinkkemper (1996, p. 276) continues that he restricts “the term methodology to scientific theory building about methodical information systems development. ... there is just one methodology of information systems development and that all research activities in this field contribute to this methodology.” Following these definitions this thesis focuses on methods and does not discuss methodologies further.

Other terms related to methods include practices, guidelines, processes, techniques, and tools. A viable definition for a *practice* is the following one: “A software engineering or management activity that contributes to the creation of the output (work products) of a process or enhances the capability of a process” (ISO/IEC TR 15504-9 1998). Notice that in the present work the term *practice* is used both when it is reported how activities *are done* in practice and, on the other hand, when it is described how an activity *can be done* in practice. A higher level view on practices is provided by *guidelines* that suggest what should be done and instructions or advice can be given how to actually implement a guideline (cf. Sommerville and Sawyer 1997, p. 3). In the present work, the term *process* follows the ISO definition: “A set of interrelated activities, which transform inputs into outputs” (ISO/IEC TR 15504-9 1998). It is important to notice that a process is considered as a part of practices here since a process has also been suggested to be “the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product” (Fuggetta 2000, p. 28). A *technique* is suitably defined by Brinkkemper (1996, p. 276): “A technique is a procedure, possibly with a prescribed notation, to perform a development activity.” Brinkkemper further claims that a technique should embody both the representational and procedural aspects of development. Finally, a *tool* can be simply defined in the following way (Brinkkemper 1996, p. 276): “A tool is a possibly automated means to support a part of a development process”.

In the present study, the technological innovation adoption process will be considered from two different perspectives and the terms related to these perspectives are defined here. The first of these is the innovation decision process that has been suggested to consist of five different stages (Rogers 1995, p. 162):

1. *Knowledge*: the adopter (an individual or other decision-making unit) is exposed to an innovation’s existence and gains some understanding of how it functions.
2. *Persuasion*: the adopter forms a favorable or unfavorable attitude toward the innovation.

3. *Decision*: the adopter engages in activities that lead to a choice to adopt or reject the innovation.
4. *Implementation*: the adopter puts an innovation into use.
5. *Confirmation*: the adopter seeks for reinforcement of an innovation-decision already made, or reverses a previous decision to adopt or reject the innovation if exposed to conflicting messages about the innovation.

The second perspective is provided by Garcia (2002) which describes technology adoption concepts exemplified in a context of large process improvement efforts and summarizes the adoption stages as a diagram shown in Figure 6. The suggested stages from *contact* to *institutionalization* complement the more general stages as suggested above by Rogers (1995) and provide a more suitable model for the present work.

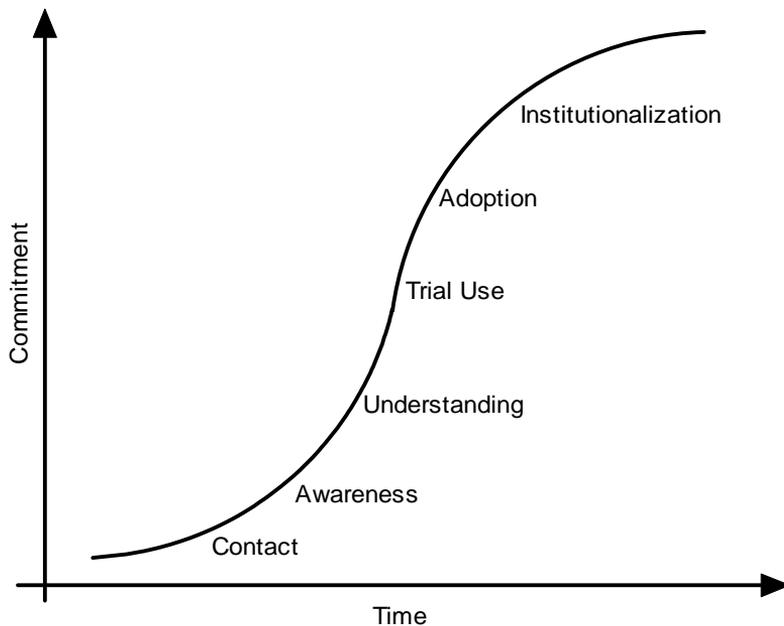


Figure 6. Notional commitment curve for adopting technological innovation (adapted from (Garcia 2002, Figure 6))

In the context of small organizations and small process improvement efforts described in the present study, the stages in Figure 6 are defined as follows. *Contact* means that a potential adopter is introduced to the idea of process maturity and improvement in general. *Awareness* refers to the stage when the potential adopter learns about shortages of the current practices, their possible consequences and, on the other hand, possible ways to solve the problems. In the *understanding* stage, the adopter learns about an innovation that is expected to prove valuable to him/her to the degree that a decision can be made about its adoption. The *trial use* stage means trying out the innovation in the adopter environment in order to get enough understanding and knowledge for the actual adoption decision. *Adoption* refers to the actual implementation of the innovation in the adopter environment to the degree that it represents a

normal way of working in the organization. Finally, the *institutionalization* (Software Engineering Institute 1995, p. 10) means that an organization has built an organizational infrastructure and culture that support the use of innovation to the degree that it is the ongoing way of doing business, even after those who originally defined them have gone. Since the *adoption* stage is of specific interest in the present work, four additional levels are defined within it:

Defined means that a solution to a problem has been developed but it might not be published yet. For example, “the standard location for documents is defined” means that an organizational level decision has been made about how/where documents are located in every project (e.g. a folder hierarchy in a specific computer).

Available means that something is defined and made accessible so that a knowledgeable user can access the artifact; for example, a document template is defined and put in a defined location in the company intranet to wait for users.

Trained means that people have been given training and support to increase their level of understanding to the degree that they know how to work on their own. For example, requirements writing and use cases training could be provided before the user is expected to develop a requirements document.

Adopted means that the solution (software, documentation, etc.) has been purchased and installed in the company and it is used in the company on a normal basis.

The last two terms introduced in this chapter are *easy to adopt* and *ready-to-use*. Both of these terms will be defined in more detail in Chapter 5 so only working definitions are provided here. In the context of the present study the term *easy to adopt* is used to refer to two aspects of a technological innovation: the technical readiness-to-use of the innovation and the need for resources in the adoption phase, e.g., to learn to use the innovation efficiently. Since the learning aspect is a common problem in everyday life (e.g., the problems of operating a video cassette recorder (Norman 1990, p. 81)) the technical readiness-to-use is defined first. The Oxford English Dictionary includes numerous explanations for the term *ready*, including the following ones (Oxford University Press 1989):

- That has passed, or has been brought, into such a condition as to be immediately likely or liable (*to do something*).
- Of action or capacity for action: Distinguished or characterized by promptness or quickness.
- In the condition of having been prepared or put in order for some purpose.
- So placed or constituted as to be immediately available when required or wished for; close at hand; handy, convenient for use.

Ready-to-use is also included in this dictionary as one variant of ready: “Construed with infinitive or with *for* and following noun, and used as attributive phrases expressing preparedness for the action indicated, as *ready-to-eat*, *-use*, etc.” Despite of the existing dictionary definitions it is clear that readiness-to-use is not unambiguous but rather subjective for each adopter and relative to the task at hand. However, a piece of hardware, e.g. a basic

hammer, is a banal example of a ready-to-use artifact as there is no way to adapt or configure it before use – it is, indeed, ready-to-use for the task it was designed for.

From the point of view of the present study, a realistic example of a ready-to-use artifact is a mobile phone which is basically a commercial off the shelf product that can simply be taken into use (Figure 7). In reality, things are not that simple. For example one leading mobile phone manufacturer seems to include in their standard product package three pieces of hardware: the actual mobile phone, a battery, and a charger. Thus a battery must be slipped into the phone, which is not always easy, and charged before the phone can be used. Still, before any calls can be made (except for emergency calls), a subscriber identification module (SIM) card must be acquired and slipped into the phone. For a first time user without specialist help it can easily take an hour or more before the mobile phone can be used for calling and receiving calls. However, if the phone is purchased from a knowledgeable dealer with a strong customer service attitude, the salesperson can change the battery to a charged one and make the phone ready-to-use in a minute or two. This technical readiness-to-use is still relative since even though calls can be made and received fairly quickly, a standard fixed-line telephone is technically ready-to-use as soon as it is plugged in the telephone network (and contractual aspects are handled). Further, in the case of a more advanced mobile phone, there may be things like multimedia messages, fast data transfer features or alike that need to be configured before they can be used. Again, as of now there is quite effective support available to make these settings in a mobile phone and an expert can do these kinds of tasks quickly and reliably. For a first time user these tasks are, however, not always very straightforward.



Figure 7. A ready-to-use mobile phone

Even though an artifact may be technically ready-to-use, a user is likely to need to learn to use it. Again, in the case of a hammer this task is fairly straightforward even though it may also prove painful. In the case of a mobile phone the learning is likely to be a physically easier but a mentally more demanding exercise. For example, in a fixed-line telephone the line must first be opened by lifting the receiver and a dial tone heard before the number is selected, but with the mobile phones this is reversed – the number must be selected first and only thereafter the actual call is initiated. There are also numerous different approaches to lock and unlock the buttons of the phone and the more advanced the phone gets, the more learning the use of the phone requires (e.g. phones with cameras, the use of a phone as a modem for a computer, sending multimedia messages, etc.). Therefore, from the point of view of the present study, a ready-to-use artifact is expected to require some installation or adaptation work before it is indeed ready-to-use. The ease of adoption, on the other hand, also takes into account the fact that a potential

adopter of the artifact may need to invest in learning (or more generally in purchases, adaptations, or alike) to be able to utilize the artifact at all and, especially, to be able to utilize it to its full potential.

3. LITERATURE SURVEY

This chapter discusses software and requirements engineering to provide a context and requirements for the method development. Section 3.1 starts by looking at studies on software development projects and how their success can be related to requirements, Section 3.2 moves on to survey research areas of interest in general, and Section 3.3 closes this chapter by discussing studies on technical issues related to the solution development of the present study.

3.1. Software Projects and Requirements

This section looks at studies reporting on software development projects and how the project success can be related to requirements. The studies of interest are the so-called CHAOS Report (The Standish Group 1994), a software project risk identification framework (Keil et al. 1998), and a study on software project success factors from the RE point of view (Hofmann and Lehner 2001). This section is closed with a summary of these studies.

3.1.1. The CHAOS Report

The reason for the interest in improving the requirements engineering practices is the sad track record software development projects have in industry. The first paper to really unveil software project failures was The CHAOS Report that analyzed over 8,000 projects and estimated the total failure costs in the United States (US) in 1994 (The Standish Group 1994). In addition to actual failure descriptions, the study also suggested project success, challenged, and impaired factors. Next a look closer at The CHAOS Report results is provided, and they are compared with other studies both from the project failure and success points of view.

The Standish Group published The CHAOS Report on the state of the industry in information technology (IT) development projects in the US in 1994 (The Standish Group 1994). It included 365 respondents representing 8,380 projects. The report estimated that in 1995 there were about 175,000 IT application development projects in the US using \$250 billion. The American companies and government agencies were estimated to spend \$81 billion on cancelled software projects and an additional \$59 billion on projects that would be completed but exceeded their original time estimates.

The CHAOS Report divided the projects into three classes: successful, challenged, and impaired ones. The successful projects were completed on time and on budget with all the features and functions implemented as initially specified. The challenged projects were also completed and operational but over the budget, over the time estimate, and offered fewer features and functions than was originally specified. Finally, the impaired projects were ones that were cancelled at some point during the development cycle. The overall success rate for the considered projects was only 16.2%, the challenged projects accounted for 52.7% of the projects, and the rest 31.3% of the projects were cancelled. Considering the challenged projects on average, only 61% of the originally specified features and functions were available. One of the key reasons for the time and cost overruns was project restarts, which meant that a project was terminated prematurely and started again. In the study there were 94 restarts for every 100 started projects. However, this does not mean that almost every project had been stopped at some point, since there were projects that were stopped many times. At the time the survey was

conducted, there were 3,682 projects running and 431, or 12%, of these were on time and on budget.

The original CHAOS Report has been followed by many respective studies. The Standish Group has done many follow-up studies and even published two CHAOS Chronicles (The Standish Group 2001). Other similar studies include the Cutter Consortium’s survey on project management with 136 information systems (IS)/IT managers from North America, Europe, Australia/the Pacific region, and Africa (Cutter Consortium 2001a). Comparing this and their earlier surveys the Cutter Consortium claims that the trend is towards smaller IT projects in terms of cost, size, and duration. Based on the available data, it is possible to compare the results of The Standish Group and the Cutter Consortium studies to see a general change in the situation. According to the Cutter Consortium study, almost all cost and time overruns were limited to 50%, while The CHAOS Report had a much more even distribution of over 200% overruns (see Table 3; the table includes only successful and challenged projects from The CHAOS Report). The average cost of an overrun in The CHAOS Report was 180% and the average time of an overrun was 222% of the original estimates. These kinds of projects that far exceed the planned time and cost estimates are often called runaway projects (Ahituv et al. 1999).

Table 3. Cost and time overruns

Percentage of overrun	Cost overrun-%		Time overrun-%	
	Chaos'95	Cutter'01a	Chaos'95	Cutter'01a
0	16	20	16	18
1-50	25	68	12	73
51-100	16	8	11	5
101-200	5	2	19	3
>200	7	2	6	0

One of the key results of The CHAOS Report is the suggested project success, challenged, and impaired factors. The three most important factors for each project class are quite similar to each other, as can be seen in Table 4. This table also shows the important role requirements play in a project outcome.

Table 4. Project success, challenged, and impaired factors (The Standish Group 1994)

Success factors	Challenged factors	Impaired factors
1. User involvement	1. Lack of user input	1. Incomplete requirements
2. Executive management support	2. Incomplete requirements and specifications	2. Lack of user involvement
3. Clear statement of requirements	3. Changing requirements and specifications	3. Lack of resources

These results are comparable with the findings of the Ginzberg (1981b) study on recurrent issues in the management information system implementation process. Ginzberg collected data

from 35 users of 27 information systems. Each user assessed 72 statements describing the implementation of the system, and an analysis of this data revealed six recurrent issues in the implementation process (Ginzberg 1981b, p. 49):

1. Extent of project definition and planning.
2. Organizational commitment to the project.
3. Breadth of analysis.
4. User responsibility for system.
5. Commitment to change.
6. User ownership of system.

A further analysis of these issues revealed that paying special attention to three of these issues was likely to increase the management information system implementation success. These issues were 1) gaining commitment to the project, 2) gaining commitment to any changes necessitated by the new system, and 3) assuring that the project is well defined and the plans are clearly specified (Ginzberg 1981b, p. 55). Ginzberg notes that these factors are quite similar to more commonly used terms like management support and user involvement, but they are not identical. This claim is supported by the Bean et al. (1975) study that found one level of support, top management selecting projects, to have no impact on implementation success and failure. The Ginzberg study, however, has obvious limitations linked to the use of a Kolb/Frohman model of change (Kolb and Frohman 1970). The study is bound by the model used and in the case the implementation process does not follow this model, the applicability of the results is questionable. This possibility is not only a theoretical one since internet software engineering has been reported to have a class of their own processes (Ramesh et al. 2002; Baskerville et al. 2003). Another problem is that the role of requirements is not clear in the study. The *extent of project definition and planning* is stated to cover “detailed consideration of organizational needs, project impacts, ..., the roles of the project team members” (Ginzberg 1981b, p. 54) which includes both requirements for the system (organizational needs) and project issues. This makes it impossible to differentiate between the roles of project management and requirements engineering in this early study.

Ginzberg (1981a), on the other hand, reports on a longitudinal study of user expectations as predictors of project success and failure. According to Ginzberg, the results of the study strongly suggest that users with realistic expectations prior to implementation are more satisfied with the system and use it more than users with unrealistic pre-implementation expectations. These findings justified proposing the involvement of users early on in the system development process and emphasizing the role of communication between users and developers to create realistic user expectations of the system. Based on the CHAOS Report, these proposals were justified over a decade later even though the reported study covered experiences only from a single system project developing a decision support system with a waterfall like process model: “for most stages, the activity of the preceding stage must be completed before the new stage can be started” (Ginzberg 1981a, p. 460).

3.1.2. Software Project Risks

The project failures raised the awareness of project risks and, for example, a software project risk identification framework was published shortly after The CHAOS Report was published (Keil et al. 1998). This study was conducted as a Delphi survey with 41 experienced project managers in the US, Finland, and Hong Kong. These experts identified the same 11 risk factors for software projects, which warranted the researchers to conclude that this is a universal set of risk factors (Table 5).

Table 5. Eleven universal project risk factors in the order of relative importance (Keil et al. 1998, p. 78)

1. Lack of top management commitment to the project.
2. Failure to gain user commitment.
3. Misunderstanding the requirements.
4. Lack of adequate user involvement.
5. Failure to manage end user expectations.
6. Changing scope/objections.
7. Lack of required knowledge/skills in the project personnel.
8. Lack of frozen requirements.
9. Introduction of new technology.
10. Insufficient/inappropriate staffing.
11. Conflict between user departments.

The proposed risk categorization framework forms a 2*2 grid presented in Figure 8. Notice that both high importance risk quadrants are closely related to requirements engineering. The *Customer Mandate* quadrant includes risk factors like lack of top management commitment to the project, failure to gain user commitment, inadequate user involvement, and the management of customer expectations, while the *Scope and Requirement* quadrant includes factors like misunderstanding the requirements and lack of frozen requirements. The *Execution* quadrant includes the traditional project execution risks and can be characterized with the question “Given what I know about the project scope and requirements, do I have a team in place that can successfully execute this project?” (Keil et al. 1998, p. 81). Examples of the relevant risk factors include insufficient/inappropriate staffing, the introduction of new technology, and the lack of required knowledge/skills in the project personnel. Finally, the *Environment* quadrant covers the risk factors that come from the project environment both inside and outside the organization. Examples of the relevant factors are changing scope/objections and conflicts between user departments.

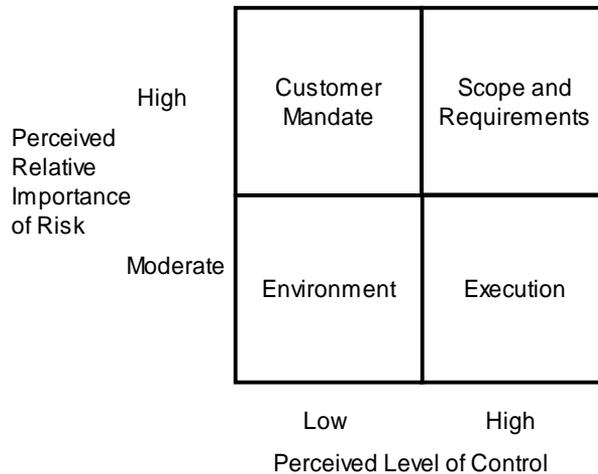


Figure 8. A risk categorization framework proposed by Keil et al. (1998, p. 80)

There are also other articles summarizing software project risk factors but only Keil et al. (1998) is reported here as an example of original research. Other articles based on previous research and anecdotal evidence can be found, for example, in Reels (1999) and Field (1997).

3.1.3. Software Project Success Factors

Software project success factors have also raised interest, and Hofmann and Lehner (2001) approach it from the RE perspective. The goal of their study was to identify the RE practices that can be clearly considered as software project success factors. The study included fifteen RE teams, including six commercial-off-the-shelf (COTS) and nine customized application development projects with recently released critical business applications in telecommunications and banking industries. The studied projects were completed on average in 16.5 months with an effort of approximately 120 person-months. The project performance was measured in three areas: quality of the RE service, quality of RE products, and process control. The study approached RE from an integrated viewpoint and investigated how team knowledge, allocated resources, and deployed RE processes contributed to project success. As a result of the study, Hofmann and Lehner propose ten best practices for successful RE teams (Table 6).

The article by Hofmann and Lehner (2001) does not provide detailed information about the conducted study. For example, it can only be assumed that the customers of these projects were large companies that could well require quality systems and accreditations from their suppliers or some specific level in, for example, SEI's software capability model. Also, the results like knowledge in the RE process in use and different aspects of the RE processes used are reported based on a 7-point Likert scale. Unfortunately, it seems that the figures shown report each aspect of the process for all the 15 teams which means that possible differences between the best and worst teams disappear in the single value reported – all the reported results lie between 4.2 and 5.4. Thus this article provides little chance for more detailed analysis of results, and the reported results are best used as they are provided.

Table 6. The best RE practices in most successful RE teams (adapted from Hofmann and Lehner 2001, p. 65)

Focus area	Best practice	Key benefit
Knowledge	Involve customers and users throughout RE	Better understanding of “real needs”
Knowledge	Identify and consult all likely sources of requirements	Improved requirements coverage
Knowledge	Assign skilled project managers and team members to RE activities	More predictable performance
Resources	Allocate 15 to 30 percent of total project effort to RE activities	Maintain high-quality specification throughout the project
Resources	Provide specification templates and examples	Improved quality of specification
Resources	Maintain good relationships among stakeholders	Better satisfy customer needs
Process	Prioritize requirements	Focus attention on the most important customer needs
Process	Develop complementary models together with prototypes	Eliminate specification ambiguities and inconsistencies
Process	Maintain a traceability matrix	Explicit link between requirements and work products
Process	Use peer reviews, scenarios, and walk-throughs to validate and verify requirements	More accurate specification and higher customer satisfaction

3.1.4. Summary

The reviewed studies indicate that requirements are well represented as project risk and failure factors. The top ten challenging and impairing factors for projects in The CHAOS Report (The Standish Group 1994) include altogether six requirements-related factors, as does the risk identification framework of Keil et al. (1998). Table 7 shows the seven requirements-related risk factors that were obtained by combining the Keil et al. study and The CHAOS Report results.

Table 7. Key requirements-related risk factors for software projects

- | |
|--|
| 1. Misunderstanding the requirements. |
| 2. Lack of adequate user involvement. |
| 3. Failure to manage end user expectations. |
| 4. Changing scope/objections. |
| 5. Lack of frozen requirements. |
| 6. Conflict between user departments. |
| 7. Incomplete requirements and specifications. |

On the success side there is no data for such straightforward comparison. The requirements-related success factors in The CHAOS report are user involvement, a clear statement of requirements, realistic expectations, and a clear vision and objectives, but a reference point for them is missing since the Hofmann and Lehner study (2001) concentrated on success factors within RE. However, comparing the Hofmann and Lehner success factors with the risk factors in Table 7 it can be claimed that the identified success factors address all these risks: risks 2, 3, 4, and 5 are addressed by the first success factor (Involve customers and users throughout RE); risks 3, 4, and 7 by the second success factor (Identify and consult all likely sources of requirements); risks 1, 4, and 5 by the eighth success factor (Develop complementary models together with prototypes); and risk 6 by the sixth success factor (Maintain good relationships among stakeholders).

The above-mentioned failure studies concentrated on large and complex projects, but there are studies suggesting that such failures are not isolated incidents but seem to happen with some regularity in companies of all sizes (Ewusi-Mensah and Przasnyski 1991; Ewusi-Mensah 1997). It is also important to notice that most software developers are found to be concerned about their current software development practices and are receptive to change (Humphrey 1998, p. 217; Nishiyama et al. 2000, p. 578).

3.2. Research Areas of Interest

In this section, previous research efforts interesting with regard to the present study are reviewed. The primary context for the work is requirements engineering but numerous other research areas were found interesting and important for the present study. The corner stones of this study lie upon the software engineering, information systems, and systems engineering disciplines from which more detailed research areas have emerged. In the following subsections requirements engineering, agile software development, domain engineering, method engineering, technology transfer, and process improvement studies are reviewed to establish a solid basis for the present study. Before going into these more detailed accounts, a brief note on systems engineering is provided.

The reason to refer to systems engineering is its emphasis on the customer. Systems engineering is by definition a general discipline as a system covers hardware, software, processes, and people. This generality is also becoming a restricting factor for systems engineering as many people find systems engineering too general a discipline to complement software engineering, and many other disciplines have been proposed for the task. Two such proposals are “software system engineering” that was brought up in the early 1980s (Thayer 1997) and “engineering of computer-based systems” a decade later (White et al. 1993). The software systems engineering approach is seldom mentioned today, so it seems to have become a marginal discipline. The engineering of computer-based systems, on the other hand, is still an active research area with some RE activity (e.g. McPhee and Eberlein 2002). On the practical side, systems engineering is closely related to RE since system requirements are the basis of software requirements specification and, consequently, also software development. This situation is evident in industrial standards like *System Definition: Concept of Operations (ConOps) Document* (ANSI/IEEE Standard 1362 1998), *IEEE Guide for Developing System Requirements Specifications* (ANSI/IEEE Standard 1233 1998), and *IEEE/EIA Standard for Information Technology - Software life cycle processes* (IEEE/EIA 12207.0 1996). For example, the ANSI/IEEE Standard 1362 states that “A ConOps is a user-oriented document that describes

system characteristics for a proposed system from the users' viewpoint" (p. i) and "the software ConOps document may be a separate document or it may be merged into the system level ConOps document" (p. iv).

3.2.1. Requirements Engineering

This section approaches RE literature from three viewpoints. First, a quick look is taken at the history of RE, then state-of-the-art reports in RE are reviewed, and finally the section is closed with research efforts closely related to the present study.

History. The term requirements engineering does not have such a clearly recorded first time use as, e.g., software engineering. As is often noted, the term software engineering was first used in 1968 as the title for a NATO conference focusing on the need for a new discipline to help manage the ever growing software system development projects (Marciniak 2002). However, according to Mack Alford's posting in the Software Requirements Engineering mailing list on December 13th, 1995, (reposted by Didar Zowghi on February 23rd, 2001) he was one of the first persons to put the words requirements and engineering together in 1973 for a program called Software Requirements Engineering Program¹. This program developed the Software Requirements Engineering Methodology (SREM) and the papers discussing it are among the first publications on requirements engineering (Alford 1978; Alford 2002). Other documented events around that time include a special session on software requirements engineering in the 2nd International Conference on Software Engineering (ICSE) in 1976, which was followed by a special issue of IEEE Transactions on Software Engineering in January 1977 covering some of the key papers from ICSE-2.

During the past 30 years a lot has happened in the RE field. The tenth anniversary requirements engineering conference was held in 2002 and two biannual conference series were united with it (International Symposium on Requirements Engineering and International Conference on Requirements Engineering). There is a journal called Requirements Engineering that is dedicated to the topic, and requirements-related articles are published frequently in journals and magazines like IEEE Software, IEEE Transactions on Software Engineering, and Communications of the ACM. The total number of requirements related articles is hard to estimate, but one fairly comprehensive requirements bibliography contains over 1,150 references to requirements-related articles prior to 2002 (Davis 2002a). In 2002, there were over 30 books focusing on requirements, a few workshops, and papers on different conferences on RE. In short RE appears a well established research area today.

State-of-the-art in RE Research. The Hsia et al. (1993) article reporting on the state-of-the-art in RE research reflects the increasing interest in RE in the early 90s. However, for the present study this review provides little encouragement, as introduction level RE is not really mentioned there. The paper does suggest a need for research into methods but as further elaboration on the topic leads to requirements notation standards and translation needs, a match with the present study idea is missing. A need to study unified frameworks appears to match

¹ The Software Requirements Engineering mailing list was replaced by the RE-online mailing list in November 2001 (Zowghi 2002).

better with the present work even though the justification for the need is derived from the problems with requirements categorizations to functional and non-functional requirements. Namely, it is stated that a unified framework is needed to integrate requirements engineering principles and concepts and to avoid fragmentation. The Zave (1997) RE research effort classification scheme, on the other hand, enumerates many promising research directions and closes the article with an observation that application domains are among the neglected dimensions in the RE research. Zave explains this topic with the A-7 method (Heninger 1980; Parnas and Clements 1986; van Schouwen et al. 1993; Parnas and Madey 1995) that was used in the A-7 aircraft specification work by stating that it “is a comprehensive requirements method for real-time process-control systems. It attempts to solve (or at least alleviate) almost all requirements problems within the limits of that application domain” (p. 319).

The later state-of-the-art reports on RE research seem to avoid introduction level RE methods more effectively than the older ones. The van Lamsweerde (2000b) RE research review focuses on processes that include modeling as a common denominator and uses a complex safety critical system to illustrate current research trends, in many cases approaching RE from the goals point of view. Even the coming 25 years of research are not suggested to address methods, integrated frameworks, or introduction level RE in this article but numerous other topics for research are named. For example, the gaps between research in RE and architecture and formal specification research are found to need more research as well as defining requirements for complex customizable packages, systematic derivation of parameter settings from requirements, and define-it-yourself approaches for small scale RE involving users as the only stakeholders. The Nuseibeh and Easterbrook (2000) review on RE research devotes a section of its own to integrated RE efforts, which also includes the observation that method engineering plays an important role in designing the RE process to be deployed for a particular problem or domain. However, the introduction level topics and integrated approaches are not listed as part of the major challenges for RE in the years ahead in this article, either. Instead, topics similar to the ones van Lamsweerde proposes in his article are repeated (architecture, formal methods, and models) even though some new areas are also brought up (the gap between requirements elicitation based on contextual enquiry and more formal approaches; non-functional requirements; and multidisciplinary training for requirements practitioners).

The Davis and Hickey (2002) article questions whether RE researchers practice what they preach. In this viewpoint article (i.e., a regular section in the journal) the different areas of knowledge needed by a requirements engineer are summarized and it is claimed that the need for “knowledge of how to decide which processes, methods and tools make most sense as a function of certain aspects of the problem domain, the specific problem being addressed, the people involved, and so on” (p. 108) has been realized only recently. Consistent with this observation, Davis and Hickey suggest situational research among the more fruitful areas for RE research and provide a reference to a problem frames book (Jackson 2001) for further information. The Kaindl et al. (2002) article summarizes two conference panels on RE technology transfer and supplements them by presenting three related research efforts (Rogers 1995; Fowler et al. 1998; Morris et al. 1998). The Kaindl et al. article manifests the need for research with the complex requirements of large-scale systems, and the closest they get to the goals of the present work is in their closing paragraph where they strongly recommend more research on the economics of RE (2002, p. 121): “It remains the case that lack of concrete knowledge of what organizations can gain from applying state-of-the-art – but also time-consuming – requirements approaches is one of the major obstacles in getting research results

adopted by practitioners. The only way to gain more knowledge about the economics of RE is for researchers to work together with practitioners.”

Finally, Wieringa (2003) claims in a rather personal position paper that “RE is building a theory about the domain of a design problem” (p. 8), and takes a very strong position against method engineering in RE research in two ways. First, Wieringa states that “research never results in prescriptions” implying that a method is by definition prescriptive, and second, he claims that “**In no case can the outcome of research be a method**” (bold original, p. 12). These claims are next alleviated by raising a need to develop domain theories that can be used to ease the decision making on what kind of methods to use under different circumstances. Wieringa seems to agree with March and Smith (1995) on the part that constructive research should include a theorization part in the end that produces theories as the outcome of a research effort. However, the Wieringa position, as reported in this paper, makes the actual *build and evaluate* phases of constructive research worthless from the RE research point of view. As the constructive approach was deemed as a promising one for the present study, the whole basis of it is questionable in light of this paper. The ideas presented in the Wieringa paper also make constructive research efforts in general unbearable from an economical point of view, since considering the time and effort involved, the theorization part presents only the tip of an iceberg in a constructive research effort. In the workshop the Wieringa paper, and especially the presented part on methods, raised resistance from the other workshop participants. If, however, this established RE researcher viewpoint does represent a more general attitude among senior RE researchers, it is no wonder that the state-of-the-art reports on RE seldom mention methods, and indications of need for introduction level methods for beginners are even harder to find.

Related RE Research. There are some individual studies in the RE area that appear particularly interesting from the point of view of the present study and these are described next. The first study of interest reports results from an industrial workshop (Morris et al. 1998), the second one reports a method engineering study utilizing method requirements specifications (Gupta and Prakash 2001), and the last two ones present evaluation frameworks for assessing RE practices (Pohl 1994; Sommerville and Sawyer 1997). An experiment in expediting the introduction of RM in industry (Fowler et al. 1998) has also been conducted and reported in close connection with the RE research but due to its focus on the introduction of practices in industry, it is reported in Section 3.2.5 (Technology Transfer). It is further noted that Yamamoto et al. (1996) report experiences from a domain-based RE methodology DREM. However, as no further information on the methodology was found in addition to this article in Japanese, it is only concluded that other researchers have been working on an apparently similar problem as the one tackled in the present work.

The Morris et al. (1998) article reports results from a workshop trying to identify and prioritize industrial and technological needs in the RE area. Namely, a marginal industrial uptake from a number of RE research projects had been observed and this workshop was arranged to investigate two things concerning the industrial uptake. First, what were possible explanations for the marginal uptake and second, identifying potential mechanisms to promote the uptake of current and future research and development projects in RE. The workshop had 26 attendees from industry, research projects, and technology transfer specialists. As key problems, the workshop identified four topics: training, inherent complexity, internal business integration, and business culture. The four most important recommendations, on the other hand, were support for the discipline, increased flexibility in the way RE research projects are conducted, revised criteria for RE research project proposal acceptance, and continued industrial relevance with RE

research projects. Thus, even if the paper elaborates the identified topics to some extent, the outcome of the workshop is not very concrete and helpful from the point of view of the present study. A closer look at the Gupta and Prakash (2001) study proves the paper to have a similar problem. The paper describes the development of an abstract language for specifying method requirements with details on a translation process from an abstract metamodel, a high level method specification, to a technical metamodel of the method. Thus, even if the study is a fine example of situational method engineering, its technical focus in automated method engineering is way too detailed to be useful for the present study.

The two last research efforts present RE evaluation frameworks reported in literature. The first one is the Three Dimensions of RE, which is a general framework for RE that is derived from the main goals of RE (Pohl 1994):

- Gaining a complete system specification out of the opaque views existing at the beginning of the process, according to the standard and/or guidelines used (Pohl 1994, p. 247). As an example of a complete system specification, the ANSI/IEEE Standard 830 (1998) is suggested even though the standard itself states that it describes “the contents and qualities of a good software requirements specification” (p. 1); other standards and guidelines describing what a final requirements document should look like can be found in, for example, Dorfman and Thayer (1990).
- Offering different representation formats, supporting the transformation between the representations (e.g. informal to semi-formal, informal to formal), and keeping the various representations consistent (Pohl 1994, p. 248).
- Allowing various views and supporting the evolution from personal views to common agreement on the final specification (Pohl 1994, p. 249).

According to Pohl, these three goals represent the fundamental dimensions of RE which are specification, representation, and agreement (Figure 9). These three dimensions can be used to classify and clarify the support methods and tools provided for RE and thus they can also be used to guide the development of new methods.

The other assessment framework (Sommerville and Sawyer 1997) is an RE process maturity model, which provides RE with a similar maturity model as the software capability maturity model (CMM) provides for software engineering (Software Engineering Institute 1995; Software Engineering Institute 2002). This maturity model was developed as a part of a REAIMS project (Sommerville and Sawyer 1997; Sawyer et al. 1999) and thus it is called here the REAIMS model. The REAIMS maturity model has three levels, which are comparable with the first three levels of the CMM model: initial, repeatable, and defined. The REAIMS assessment process includes four phases: selecting people to interview, initial scoring, refinement, and final maturity level calculation. In the scoring phase, current practice is checked off against the REAIMS guidelines and use of each guideline is assessed on four levels (Table 8).

The guidelines are classified on three levels (basic, intermediate, and advanced) and separate sums are calculated for each category by summing up the numerical scores. This information is then used to select the appropriate maturity level from initial, repeatable, and defined levels. There are altogether 66 guidelines that should be considered in the assessment, but Sommerville and Sawyer also suggest a top ten list of guidelines which should be applied by all organizations (Sommerville and Sawyer 1997, p. 31).

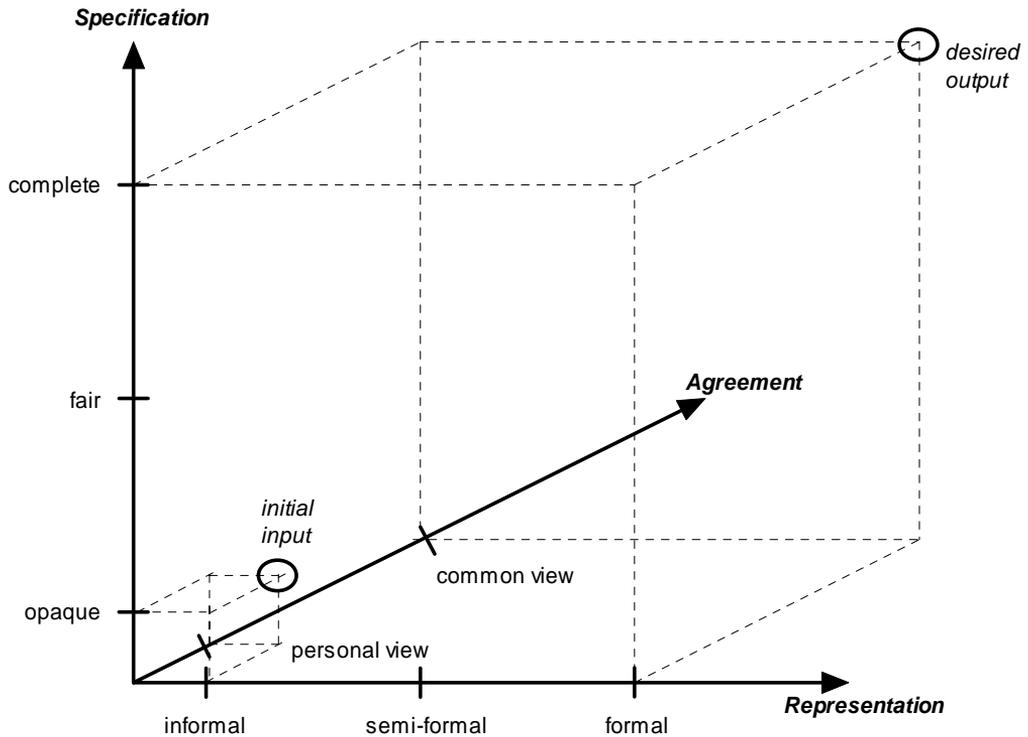


Figure 9. The three dimensions of requirements engineering (Pohl 1994, p. 249)

The practical relevance of the presented frameworks is not easy to assess but a look at literature referring to these models provides some indication of it. A fair literature search found only one experience report on the use of the Three Dimensions of RE framework in practice (Matulevicius 2004), and Professor Pohl himself was not aware of such articles by the fall of 2003 (personal communication). The REAIMS model, on the other hand, appears to experience increasing interest among researchers and it has been utilized in a number of research efforts. For example, one project has reported their use of REAIMS as a basis for RE process improvement efforts (Kauppinen and Kujala 2001; Kauppinen et al. 2002) and one individual paper reports experiences from extending extreme programming with RE (Nawrocki et al. 2002).

Table 8. Different REAIMS use levels, respective scores, and descriptions (adapted from (Sawyer et al. 1999, p. 83))

Use Level	Score	Description
Standardized	3	Guideline is in documented use in the organization, it is followed and checked as a part of quality assurance procedures.
Normal use	2	Guideline is widely followed in the organization but it is not mandatory.
Discretionary	1	Guideline is followed by the discretion of each project manager.
Never	0	Guideline is rarely or never applied.

3.2.2. Agile Software Development

One of the most recent software engineering trends is the movement towards agile methods. The real heyday of the agile movement started with Kent Beck's Extreme Programming (XP) book (1999b), even though most of the agile methods were developed in the mid 90's (see e.g. Beck 1999a; DSDM 2002; SCRUM 2002). The term *agile* was actually established only in the beginning of the year 2001 when the Agile Software Development Manifesto was developed by people representing Extreme Programming, SCRUM, the dynamic systems development method (DSDM), Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and other lightweight or adaptive methods (AgileAlliance 2002). Agile methods were developed as a response to a need to have an alternative to "document driven, heavyweight software development processes" (AgileAlliance 2002). This need was confirmed in a Cutter Consortium survey with 200 IS/IT managers – 54% of the respondents said that they used in-house methods that would be described as agile (Charette 2001). At the time of that survey the brand name Agile Methods was not yet in broad use, since 38% of the respondents used XP, 23% Feature-Driven Development, 22% Adaptive Software Development, and 19% Dynamic Systems Development. In general, the heavyweight methods are still dominantly in use – the Cutter survey reports that the three dominant heavyweight methods in use are the Rational Unified Process (51%), those that support the Software Engineering Institute's Capability Maturity Model (27%), and those that support ISO 9000 (26%). However, the survey closes with a result that in 2003 approximately 50% of the respondents expect 50% or more of their projects to use agile methods.

The need for agile approaches seems clear and the interest in them has also resulted in different kinds of analysis between agile and heavyweight methods. In another Cutter Consortium survey with 30 IT managers from large organizations, the key perceived benefits of using an agile method over a heavy one were faster time to market (68%), better response to customer requirements (54%), improved staff morale (36%), and more innovative solutions (25%) (Cutter Consortium 2001b). A more thorough qualitative assessment of the differences between agile and plan-driven (i.e. heavyweight) methods is provided by Barry Boehm (2002). This assessment acknowledges first the home grounds for each method type where they perform the best – and clearly better than the other type – and continues by focusing on the middle ground where both the methods need to address less than optimal project characteristics. For example, the agile methods rely heavily on excellent people, as described by Larry Constantine: "All of the agile methods put a premium on having premium people and work best with first-rate, versatile, disciplined developers who are highly skilled and highly motivated. Not only do you need skilled and speedy developers, but you need ones of exceptional discipline, willing to work hell-bent-for-leather with someone sitting beside them watching every move" (Constantine 2001). As Boehm (2002, p. 65) notes, 49.9999% of the world's software developers are below average, so this may impose a serious risk for agile methods. The plan-driven methods, on the other hand, acknowledge this situation and prepare for possible problems with documented plans and architectures that can be reviewed by external experts. Table 9 summarizes the home ground characteristics for both agile and plan-driven methods to clarify their differences. As a conclusion, Boehm (2002) suggests that both method types are needed and each project should evaluate its project risk exposure profile to be able to balance between planning and agility.

Table 9. Home ground for agile and plan-driven methods (Boehm 2002, p. 68)

Home-ground area	Agile methods	Plan-driven methods
Developers	Agile, knowledgeable, collocated, and collaborative	Plan-oriented, adequate skills, access to external knowledge
Customers	Dedicated, knowledgeable, collocated, collaborative, representative, and empowered	Access to knowledgeable, collaborative, representative, and empowered customers
Requirements	Largely emergent, rapid change	Knowable early, largely stable
Architecture	Designed for current requirements	Designed for current and foreseeable requirements
Refactoring	Inexpensive	Expensive
Size	Smaller teams and products	Larger teams and products
Primary objective	Rapid value	High assurance

A closer look at two agile methods, eXtreme Programming (Beck 1999b) and Crystal Orange (Cockburn 2002), reveals that they are software development methods where RE is just one part of the whole process and does not get treated deeply. According to Cockburn (2002, p. 175) agile methods in general try to defer documenting activities for as long as possible and preferably implement documents in a follow-up project as little as possible. The following quote describes well the general approach agile methods have to documentation (Cockburn 2002, p. 175):

... guessing how much can be bound in your group's oral tradition and how much has to be committed to archival documentation. Recall that it does not matter, past a certain point, whether the models and other documentation are complete, correctly match the "real" world (whatever that is), or are up to date with the current version of the code. What matters is whether the people receiving them find them useful for their specific needs.

The Crystal Orange method does have a requirements document as one work product (Cockburn 2002, p. 205). However, work product templates are to be developed and maintained as local standards by the development team, so there is no help for defining a requirements document template. The XP, on the other hand, has story cards that contain a name and a short paragraph describing the purpose of the story (Beck 1999b, p. 88). The primary purpose of these stories is project planning and deciding what to include in the next release, but they also provide a common understanding of the purpose of each function. Once the story cards for the next release have been agreed upon, further elaboration on the requirements is done orally.

As a final point about agile methods, the three levels of listening Cockburn (2002, p. 14) proposes for people learning and mastering new skills is presented. The first level is called the following stage and it refers to people looking for one procedure that works. The second level, detaching, refers to people who try to locate the limitations of the single procedure and look for rules about when the procedure breaks down. The third, fluent, stage means that it is irrelevant to the practitioner whether he/she is following any particular technique or not – he/she

understands the desired end effect and simply makes his/her way to that end. Cockburn also reports Kent Beck to have replied to a question about XP and the Software Engineering Institute's Capability Maturity Model with XP's three levels of maturity which are similar to his three levels of listening (2002, p. 17):

1. Do everything as written.
2. After having done that, experiment with variations in the rules.
3. Eventually, don't care if you are doing XP or not.

From the point of view of the present study, the agile software development approaches provide a complementary view for the previously predominant maturity oriented process improvement literature. The emphasis on agile, or lightweight, processes and people doing the work seems to present a natural fit for small organizations in the non-critical application domain. Finally, the reported need for personal development and discipline in work suggests that agile approaches are not quite the opposite of the maturity oriented approaches either.

3.2.3. Domain Engineering

Domain engineering refers to software development based on a priori experience or knowledge about the task and domain at hand. In this section the fundamental ideas of domain engineering and its theoretical justification are summarized since specificity was assumed to prove useful in developing an easy to adopt solution.

Domain engineering complements the recent agile trend in software engineering in two ways: it is based on reusing experience from similar projects, and the principles for this research area were suggested already in the 70's. Comparisons between software and hardware industries brought up the idea of reuse as a potential solution to the problem of increased software development costs (Braun 2002, p. 1197). Since the original problem definition included focusing on a single domain, the reuse goal can be further refined by studying domain engineering, which is defined as "the systematic process collecting, organizing, and storing past experience in building systems in a particular domain" (Czarnecki 2002, p. 433). The experience is stored in the form of reusable assets like documents, patterns, reusable models, and components (e.g. individual requirements). An additional goal for domain engineering is to provide infrastructure for reusing these assets during application engineering (i.e. the process of building new systems). The domain and application engineering research areas have recently been affiliated with product line engineering (Czarnecki 2002, p. 434) as the domain engineering approach suits especially product development where multiple products (a product family or a product line) have common properties. Although product lines are a fairly new research area, the product family idea was suggested already in 1976 by Dijkstra and Parnas. Parnas' early definition for a "program family" was the following (Czarnecki 2002, p. 433):

We consider a set of programs to constitute a family, whenever it is worthwhile to study the programs from the set by first studying the common properties of the set and then determining the special properties of the individual family members.

The research on cognitive fit provides a theoretical justification for domain engineering. This stream of research has studied the history of software application domains (Glass and Vessey 1992; Vessey 1997) even though the actual foundation of cognitive fit was laid by Vessey

(1991) studying the differences between graphical and tabular representations and the types of tasks they support. Vessey bases her research on the strong and weak problem solving methods (Newell 1969) and develops a notion of cognitive fit from them: “complexity in the task environment will be effectively reduced when the problem-solving aids (tools, techniques, and/or problem representations) support the task strategies (methods and processes) required to perform that task” (Vessey 1991, p. 220). This notion abstracts quite a bit the original idea between weak and strong problem solving methods, so Vessey and Glass (1998, p. 99) also explain them in commonsense terms: “The words *strong* and *weak* are deliberately chosen. A strong method, like a specific size of wrench, is designed to fit and do an optimal job on one kind of problem; a weak method, like a monkey wrench, is designed to adjust to a multiplicity of problems, but solve none of them optimally.” The Vessey theory was originally tested in a fairly simple and specific domain but it has been used successfully in the systems development domain (Vessey and Weber 1986) and it is claimed to suit this domain well (Vessey and Glass 1998). The general objective for striving after cognitive fit is intuitively appealing: “cognitive fit leads to an effective and efficient problem solution. If cognitive fit is lacking, it does not mean that the problem cannot or will not be solved; it simply means that the solution will be less effective and/or less efficient” (Vessey and Glass 1998, p. 100). The Vessey and Glass (1998, p. 102) implications on research also provide clear support for domain engineering: “Research, these findings suggest, should focus on systems development approaches based on cognitive fit, the matching of methods to tasks. Methods should be studied from the viewpoint of their benefit to particular classes of applications, to facilitate matching method to application task.”

Glass and Vessey (1998) report that little real results from the domain specific approaches had been achieved so far even though the need for such approaches was broadly acknowledged. However, after the publication of their article, research efforts underlining the differences between different domains (Ramesh et al. 2002; Baskerville et al. 2003) and the benefits of domain specific approaches (Port and Dincel 2001) have been reported. In light of these studies, the question of Glass and Vessey (1998, p. 188) is as topical as before: “How much more progress is possible without taking the particular type of application directly into account?”

3.2.4. Method Engineering

ME was originally proposed in 1992 as “methodology engineering” based on the observation that organizations devise, use, and adapt development methods in their information systems development projects (Kumar and Welke 1992, p. 257). As the topic started to gain more interest, the term method engineering was proposed with the following definition: “the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems” (Brinkkemper 1996, p. 276). ME can be divided into four key research areas: meta-modeling techniques, tool interoperability, situational methods, and the comparative review of methods and tools. Since a domain specific approach was chosen for the present work, the situational methods, or situational method engineering, appear to match the needs of the study fairly well from the method adopter point of view. However, in a tool-supported ME environment two additional roles are introduced in an organization: the method engineer and the method administrator who are responsible for generating the right methods for each project and for the contents of a method base respectively (Odell 1996, p. 4).

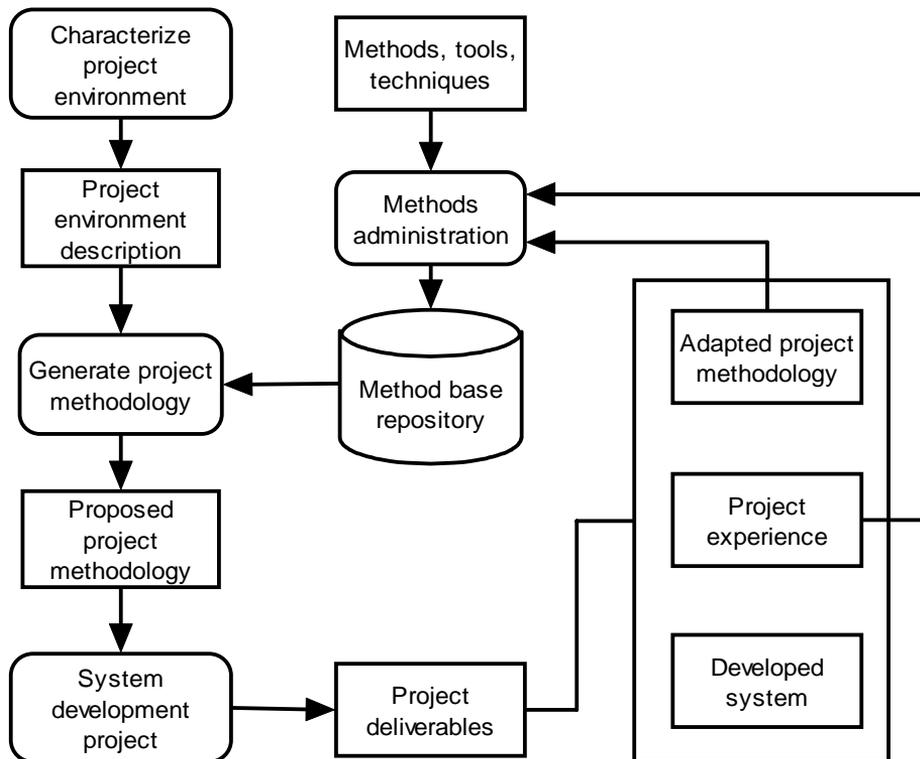


Figure 10. An object-flow diagram specifying the process of modular method construction (Odell 1996, p. 3)

Situational method engineering adapts methods to the needs of a particular project, which, in the case of IS, refers to designing, constructing, and adapting development methods for IS development (Odell 1996, p. 2). The degrees of flexibility for IS situational methods can be divided so that there is use of rigid methods, selection from rigid methods, selection of a path within a method, selection and tuning of a method outline, and modular method construction. The process of modular method construction should be supported by an automated tool. A schematic diagram of the modular method construction process is shown in Figure 10.

However, situational method engineering still has many fundamental problems to solve. The main problems are summarized in Table 10 below. ter Hofstede and Verhoef (1997) consider situational method engineering to be still in its infancy and highlight two aspects of this conclusion. First they claim that ME has focused on syntactical aspects and not on semantics, which prevents solving fundamental issues. Their second claim is that “Much more empirical research is needed to *substantiate* the claims associated with the potential benefits of situational method engineering” (*italics original*) (cf. also Tolvanen et al. 1996, p. 310; ter Hofstede and Verhoef, p. 417).

Despite their criticism, ter Hofstede and Verhoef conclude their paper with a statement that situational method engineering may well be feasible in a restricted context. As one possibility they mention focusing on methods and techniques that are suitable for specific domains only.

Table 10. Overview of the main problems identified with situational method engineering (ter Hofstede and Verhoef 1997, p. 417)

Selection of method fragments	Design at appropriate level of coherency and granularity
Storage of method fragments	Complexity of meta-modeling language
Retrieval of method fragments	Characterization of method fragments Formulation of needs
Assembly of method fragments	Support for customization of method fragments Integration of method fragments

This is not a new idea and some work in the RE domain has already been done (Rolland and Grosz 1994; Rolland et al. 1999) even though empirical evidence of the practical applicability of this approach is still limited.

Introna and Whitley (1997) provide a critique of *method-ism* which they consider as the dominant mindset in ME and define it with the following two axioms (p. 236):

- Methodology is necessary and sufficient for information systems' development success.
- If systems developers have a suitable methodology at their disposal they will use it.

Introna and Whitley consider the understanding of and involvement in the problem situation crucial to a successful use of a method. Tools (or methods) are used only if they are ready-to-hand in the world of doing; if they are not, they will break down and be ignored in the pragmatics of getting the job done (Introna and Whitley 1997, p. 237). It is further claimed that tools that are used in order to get the job done become invisible to the task (Introna and Whitley 1997, p. 238). The reasons for methodology breakdowns, on the other hand, can be classified in three categories, as summarized in Table 11.

The Introna and Whitley concluding discussion can be summed up as the following five ideas (1997, p. 243):

Table 11. Summary of the reasons for breakdown of engineered methodologies; the centrality of understanding (Introna and Whitley 1997, p. 242)

Category	Role of understanding	Example
Confines of method	Understanding of the confines of method is required to appreciate the limitations of methodologies and to know when they can be sidestepped to make progress	Variety-poor systems Lack of creativity
Contextual knowledge	Understanding is required to provide tacit knowledge to enable the methodology to be used and to encourage stability	Meaningless systems Tacit knowledge required Instability
History	An understanding of the history of a methodology helps with the discourse of domination and provides a better appreciation of the values inherent in the methodology	Discourse of domination Value-ladenness

- It is essential to abandon the mindset that a suitable single methodology is all that is required.
- It should be expected that elements from different methodologies are picked up and used as and when necessary. Methodologies and their associated tools and techniques should therefore be designed with this improvisation in mind.
- Method engineers should expect that their techniques and tools will be used in ways that they themselves did not anticipate.
- Rather than being given only a short training course in a particular methodology, it should be expected that would-be systems developers will spend considerable time learning about the use of tools in the context of information systems development, preferably through formal apprenticeship.
- Method engineers should focus on helping developers in their job of creating usable information systems by creating methodologies, techniques and tools that enable systems developers to focus on their task and develop their understanding of what they are doing, and not end up focusing on the tools that they are using.

For the present study, this analysis on method engineering and especially method-*ism* proved thought provoking. First of all the discussions on ready-to-hand tools and tools becoming invisible to the task at hand initiated a fundamental development of the desirable characteristics of methods, and second, this article provided indications both of the limitations of methods and their readiness-to-use. Thus the solution concept development of the present study (Chapter 5) was heavily influenced by this article.

As a final point about method engineering, a viewpoint from well before the establishment of ME is provided. Namely, software development tools can be categorized in three classes as follows (Riddle and Fairley 1980, p. 1):

- Cognitive tools are the problem-solving techniques used to develop software systems... They are characterized by the fact that they enhance the intellectual capabilities of those who use them.
- Augmentive tools are those which aid software development practitioners in much the same way that hammers, screwdrivers and saws aid carpenters – they increase the practitioners’ “powers.”...
- Notational tools (i.e., languages) provide the media in which ideas may be expressed and communicated and they play an essential role in our ability to formulate concepts and solve problems.

3.2.5. Technology Transfer

Technology transfer is a recurring topic in software engineering and IS literature. The problem of improving the practitioners’ working practices is, however, viewed from different angles. In the software engineering area these issues are generally addressed under the term *technology transfer*, while in the IS field *technology acceptance* and *diffusion of innovations* appear more relevant viewpoints. The basic distinction between these approaches is that it is generally hard to transfer technology or knowledge to someone that is not perceptive to adopt it and, consequently, the receiver point of view is emphasized in the IS field. In this section,

technology transfer is approached from two viewpoints – problems in introducing new methods and practices in industry and, on the other hand, factors affecting the technology acceptance. As the goal of the present study is to introduce new practices in industry, these two issues present central challenges for it.

Introduction Problem. In this section, three descriptive studies on the results of technology transfer efforts are reported. First Hardy et al. (1995) report on the use, limitations and customization of structured systems development methods in the United Kingdom. The results of this survey with 102 responding companies include the following:

- 44% of the responding departments claimed to use a recognized structured method or formal specifications.
- 38% used “in-house” methods that, according to the authors’ industry contacts, are simply informal collections of techniques.
- 88% of the departments customized the methods they used.
- 62% of the departments gained the knowledge for the customization from their own experiences, 29% from external consultancy, 5% from internal consultancy, another 5% from software packages, and 2% from books and guides.

Hardy et al. (1995, p. 467) also note that “the more broadly based, or widely supported a method is, the greater potential there is for its uptake.” However, this will result in an ever increasing size of the method with two possible consequences: firstly, the sheer size of the documentation may frighten the potential adopters from considering the method, and secondly, increasing the applicability of the method to different project types can make it too hard for systems developers to decide which parts of the method are suitable for any given project.

Another interesting study on factors affecting the completion of the requirements development stage in projects having different characteristics is reported by Chatzoglou (1997a). This was a mail questionnaire survey with 107 projects from 74 organizations on medium sized projects of mainly up to 3 years of duration, 30 employees, and a total effort of less than 192 person-months. A third of the projects in the study did not use any method at any stage of their development process, whilst 53% used some method for the development process. Of the projects using a method for development it covered also the requirements development phase in 46% of the cases. Interestingly enough, 16% of the projects had a method for requirements development but not for other development processes. The use of methods was much more common with academic, software house, and consultancy projects (between 70% and 82%), while industry applied methods only in about 30% of their projects. As possible explanations for the differences Chatzoglou (1997a, p. 632) suggests the following three reasons:

1. People who work for industry are not sufficiently informed about new methods and the way they should be used (Chatzoglou 1997b).
2. Methods are difficult to use in practice, mainly because they are not well documented or an extensive training program is required before they can be put into use (Tamai 1993; Chatzoglou 1997b).
3. Projects developed by industry are usually small sized projects which are developed for internal use. People who develop them have the impression that they do not need any method (Tamai 1993; Chatzoglou 1997b).

The general conclusions from the Chatzoglou (1997a) study are the following (p. 636):

1. There were not enough resources available.
2. The quality of tools and techniques initially employed was not adequate.
3. The management style and techniques adopted initially did not always seem to be the most appropriate ones.

The third study, (Fowler et al. 1998), reports experiences from the introduction of requirements management practices in industry. A website – a prototype transition package – that was developed to find out if the adoption of requirements management practices to major adopter categories (Rogers 1995) can be expedited with a suite of RM-specific materials for change agents². The transition package was a password protected website with about 100 documents covering both the introduction of RM practices and their performance. The website data was analyzed for a period of about three months and during this time over 14,000 hits were recorded. Unfortunately, the number of users or companies using the site is not reported and only perceptions of the website are reported in addition to survey results and usage analysis.

Fowler et al. summarize numerous related studies succinctly as follows: “Improvement in the state of the practice in software engineering depends upon improved methods of technology introduction; these methods limit or enable adoption of useful new software engineering technologies and practices” (p. 139). As the outcomes of their own study, on the other hand, Fowler et al. report that all the website users except one were new to RM and software engineering in general. Thus it is not surprising that an analysis of the study findings “implies that prototype users wanted a kind of primer for introducing RM, one that would spell out where to start and how to proceed, and one that was tailored to different organization types” (p. 143). These findings line up well with the stated need for a comprehensive, integrated, and easy to use package (p. 142) for use with minimal adaptation or as is (p. 145). As the main results of their study Fowler et al. report a uniformly positive response from the prototype users concerning both the idea of the transition package and the contents of the presented package. As concrete benefits from the package the users reported time savings and structure for the introduction effort. The prototype also generated many improvement ideas including, e.g., a better integration between the transition process model and the provided artifacts.

Affecting Factors. There are also explanatory studies on the factors affecting technology transfer. The Riemenschneider et al. (2002)³ study on the software developer acceptance of methods used five different theoretical models to explain the results. These models were the Technology Acceptance Model (TAM) (Davis 1989; Davis et al. 1989), TAM2 (Venkatesh and Davis 2000), Perceived Characteristics of Innovation (PCI) ((Moore and Benbasat 1991) developed based on diffusion of innovations (Rogers 1995)), the Theory of Planned Behavior (TPB) (Ajzen 1991), and the Model of Personal Computer Utilization (MPCU) (Thompson et

² This research effort was done as a part of software practice improvement efforts in accordance with CMM and its key process areas; the software CMM discusses only requirements management activities, not requirements engineering as a whole.

³ The results of this study are also reported in Hardgrave et al. (2003).

Table 12. Construct map (Riemenschneider et al. 2002, p. 1139)

Theoretical Models					
Construct	TAM	TAM2	PCI	TPB	MPCU
Usefulness	Usefulness	Usefulness	Relative Advantage	Attitude	Job Fit
Ease of Use	Ease of Use	Ease of Use	Complexity		Complexity
Subjective Norm		Subjective Norm		Subjective Norm	Social Factors
Affect					Affect
Voluntariness		Voluntariness	Voluntariness		
Compatibility			Compatibility		
Result Demonstrability			Result Demonstrability		
Image			Image		
Visibility			Visibility		
Perceived Behavioral Control – Internal				PBC - Internal	
Perceived Behavioral Control – External				PBC - External	Facilitating Conditions
Career Consequences					Career Consequences

al. 1991). It should be noticed that these five models were originally developed for the tool acceptance domain and in this study their applicability in the method acceptance context was validated empirically. Table 12 summarizes the constructs of these models and shows how Riemenschneider et al. found the constructs to compare with each other; the meanings of the constructs are summarized below.

The *usefulness* refers to the extent an adopter perceives the system to enhance his or her job performance while the *ease of use* addresses the perceived freedom of effort in both learning and using the system. The *subjective norm* is defined as the degree to which people assume that other people who they find important think that they should perform the behavior. The *affect* refers to the positive or negative emotional reactions an individual associates with a particular act while the *voluntariness* refers to the extent to which potential adopters perceive the action to be nonmandatory. The *compatibility* is defined as the degree to which the innovation is perceived as being consistent with the existing values, needs, and past experiences, the *result demonstrability* is defined as the extent to which the innovation is expected to demonstrate tangible advantages, the *image* is defined as the degree to which use of the innovation is perceived to enhance one’s image or status in one’s social system, and the *visibility* is the degree to which others can observe the benefits of the innovation. The *perceived behavioral control* refers to one’s perception of the internal and external constraints on performing the

behavior, and finally the *career consequences* refers to the long term consequences of the use and future pay-off for the action.

As shown in Table 12, different models use different terms for similar constructs which is well demonstrated by the terms *usefulness* and *relative advantage*. Rogers (1995, p. 15) claims that relative advantage is one of the five characteristics of innovations that can be used to explain the different rate of adoption different innovations experience. It has been defined by Rogers (1995, p. 15) as “the degree to which an innovation is perceived as better than the idea it supercedes.” This definition clearly posits a perceived difference between the current situation and the expected improvement in it. In his technology acceptance model Davis (1989, p. 320) suggests an alternative name, *perceived usefulness*, for basically the same construct: “the extent they [people] believe it will help them perform their job better.” Moore and Benbasat (1991) discuss this topic in a wider context and report deficiencies in both the definitions as reported by other researchers (p. 197-198). For example, they report the Tornatzky and Klein (1982, p. 34) critics against relative advantage: “If relative advantage is measured in terms of profitability, or social benefits, or time saved, or hazards removed, why bother to refer to relative advantage at all?” Moore and Benbasat elaborate the Tornatzky and Klein approach by estimating that it “could lead to a plethora of the scales and terms, all of which would be subsumed under the idea of *relative advantage*” (p. 198). Despite of the terminological differences, many studies (e.g. Davis 1989; Moore and Benbasat 1991) report that this basic construct has been proven to predict reliably the acceptance of IT in practice, and thus establishing a relative advantage with the anticipated solution appears important.

As the results of their study Riemenschneider et al. report that method adoption intentions were driven by usefulness, voluntariness, compatibility, and subjective norm. There is another study that was apparently conducted around the same time with a very similar goal. This study by Huisman and Iivari (2002) of individual deployment of systems development methods reports that from the perceived characteristics of innovations, the relative advantage (i.e., usefulness), compatibility, and trialability affected the deployment of methods – a result that matches the above Riemenschneider et al. study quite well. The differences between the results can be expected to be caused by the quite different settings for the studies: the Riemenschneider et al. study was conducted in the USA in one large Fortune 1000 company where all the developers were instructed to start using a custom-made method by a formal written policy statement from the chief information officer; from the 185 developers 128 responded to the study questionnaire. The Huisman and Iivari study, on the other hand, was conducted in South Africa and covered 83 organizations with 234 developers and 73 managers. Even if Huisman and Iivari do not report the organization profiles in more detail, it is apparent that the organizational and cultural differences may have caused the differences between the findings of these two studies.

It should be noted that these two studies did not study identical construct sets. The Riemenschneider et al. study has excluded the trialability construct from the PCI list (Table 12) even though this omission is not explained in the article in any way. However, Moore and Benbasat (1991) note that the inclusion of this construct can be decided by researchers on a case by case basis and since in the Riemenschneider et al. study the organization mandated the method usage and, thereby, took the responsibility and risk of method adoption, the role of trialability can be assumed insignificant. In the Huisman and Iivari study setting, on the other hand, it appears hard to control the risk of adoption in any way and inclusion of the trialability characteristic in the survey was a natural decision (Moore and Benbasat 1991, p. 210). Huisman and Iivari actually used the original diffusion of innovation characteristics as defined by Rogers

(1995), and thus they did not study image and visibility constructs. Also voluntariness was considered under the organizational characteristics since Huisman and Iivari actually included in their conceptual research model individual, task, organizational, and environmental characteristics in addition to the diffusion of innovation characteristics just discussed. From the point of view of the present study, the set of innovation characteristics seems like a suitable basis for the method construction phase. Namely, even though different studies report slightly different results as the key factors affecting adoption, there appears to be an overall consensus about the importance and validity of these characteristics.

3.2.6. Process Improvement

Technology transfer is a viable context for introducing new practices in industry but it is not the only one – another well established research area having similar objectives is process improvement. Process improvement activities can be further divided into software and business process improvement of which the latter is better known as business process reengineering. The reason to look also at business process reengineering (BPR) in the present work is that it presents a higher level activity than software process improvement (SPI). That is, software related processes are one part of business processes that must be considered when BPR efforts are conducted in houses developing software. Another noteworthy matter is that the agile software development discussed in Section 3.2.2 can also be considered as a kind of SPI approach that focuses on agile techniques.

A key difference between BPR and SPI is that BPR is in general considered as a radical approach to change (e.g. Hall et al. 1993; Hammer and Champy 1993) while SPI is usually suggested as an incremental or iterative activity. Hammer and Champy (1993, p. 32) define BPR as follows: “Reengineering is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service, and speed.” This approach differs quite a bit from the SPI approach suggested by Sommerville (2001, p. 559): “Process improvement is a long-term, iterative process. Each of the activities in Figure 25.2 [Analyze process, Identify improvements, Introduce process change, Train engineers, and Tune process changes] might last several months.” However, these general approaches are challenged by more specific research reports. For example Caron et al. (1994) and Harkness et al. (1996) report on BPR efforts that started small and only later achieved larger scales or more complex parts of the companies through iteration and increments. In the same vein SPI efforts can be radical. Riemenscheider et al. (2002, p. 1141) report that “compared to tool adoption, methodology adoption is a more fundamental and radical change to the behavioral processes in conducting one’s work.” Considering, for example, tools for change management, defect tracking, system design, regression testing, or version management it is clear that there is little chance for a grace period when the tools are used only occasionally without risking the benefits of the tool use. Thus a fundamental change of practices may be needed at the time of a tool adoption and, on the other hand, adopting a new method may impose similar requirements on radical changes in the work practices. In the present study, the need for both radical and incremental changes is acknowledged and therefore both BPR and SPI literature are surveyed next. The BPR literature is summarized first and the SPI literature is then discussed in three parts: SPI related problems, success factors, and case studies.

Business Process Reengineering. The business process reengineering literature includes at least three kinds of papers that provide interesting higher level reflections on process improvement efforts. These are in-depth case studies reporting experiences and lessons learned from individual companies (e.g. Caron et al. 1994; Harkness et al. 1996), high level analysis of BPR efforts and their success factors (e.g. Hall et al. 1993; Huizing et al. 1997), and in-depth reports on empirical development of reference frameworks for BPR efforts (e.g. Kettinger et al. 1997). These studies are not introduced in more detail here since their wider scale applicability to the present effort is not self-evident but, instead, references to them will be provided later in the work. However, these studies are important for the present study in two ways. First, they demonstrate similar process improvement efforts in another discipline providing opportunities for learning from others' experiences and second, these studies represent in many cases thoroughly reported major research efforts that provide a comprehensive and complete picture of the conducted work. As the following discussion on SPI efforts will show, this is not always the case in software engineering research.

SPI Related Problems. The Beecham et al. (2003) study on SPI problems in twelve software companies reports as the main finding an association between the company's capability maturity and patterns of reported problems. The companies with low maturity were busy dealing with project related problems like documentation, timescales, tools, and technology while companies with high maturity were more concerned with organizational problems. The second finding in this study is that different practitioner groups (i.e., developers, project managers, and senior managers) have different concerns. Beecham et al. report that developers are mostly experiencing problems with requirements, testing, documentation, communication, tools and technology; project managers are concerned with timescales, change management, budgets and estimates; and senior managers report the most problems with goals, culture, and politics. Even though the results of this study are very interesting, Beecham et al. list several limitations to the study. First of all, the sample of the study included 12 companies, 49 focus groups, and 200 staff members but only one of the organizations was at CMM level 4 and no company was at level 5. The focus groups, on the other hand, did not have consistent representation from different staff groups in different companies. Finally, in the study all the problems had equal importance – which is unlikely to be true in a commercial setting – and not all the companies are likely to have the same problems found in the this study. Thus generalizing the problems to other companies should be done with appropriate caution.

Two articles by Watts Humphrey (1998; 2002) report SPI related problems he has observed along his long career in this area. For example, he states the following (Humphrey 1998, p. 202):

Briefly, the state of the software practice has been so bad for so long because:

- The educational system does not provide graduates with the practical skills they will later need. In addition to technical knowledge, they need to make accurate plans, to measure and manage the quality of their work, and to function effectively as development team members.
- Few software organizations are willing or able to provide the remedial training their new engineers need.

- Today's software organizations have few if any role models who consistently demonstrate effective work habits and disciplines.

As the most critical issue to improve the current state he suggests "getting the software engineers to use improved methods" (Humphrey 1998, p. 202). Based on his experiences with Personal and Team Software Processes, Humphrey reports that engineers generally follow clearly structured sets of tasks and defined processes for handling them (Humphrey 1998, p. 216). However, he notes that at least students need far more explicit process guidance to consistently use sound methods than has been provided in the past (Humphrey 1998, p. 216).

In his later article, Humphrey provides an even broader view on his work. The two central observations concerning his work with the CMM are the following. First, CMM was developed to address three key concerns: to get management attention to process related problems, to maintain a long-term improvement focus, and to guide the improvement work (Humphrey 2002, p. 47). Second, the reason CMM does not consistently or predictably change engineering behavior is that it lacks specific guidance (Humphrey 2002, p. 53). To tackle this issue Humphrey developed the Personal Software Process (PSP) (Humphrey 2002, p. 54): "I realized that to get people to work this [CMM level 5] way, we would have to show them precisely how to do it." As an explanation to the fact that PSP has not gained wide-spread use and that even people who have used it tend to stop using it, Humphrey says "disciplined behavior is not normal or natural to most people. It requires extensive training, a supportive environment, and a change in management style" (Humphrey 2002, p. 59). Despite the problems in getting the Personal and Team Software Process methods in large scale use in industry, Humphrey still believes in methods and states that "Method is critical and the disciplined practice of sound methods is the only way to learn to do consistently high quality work" (Humphrey 2002, p. 66).

A final viewpoint on SPI problems is taken from the Nishiyama et al. (2000) article summarizing the problems faced in introducing technology in the company. The tight schedules and budgets have led to a situation where "developers, as well as organizations within the corporation, cannot readily afford the time or money needed to learn and incorporate new techniques and technologies in their development practices" (Nishiyama et al. 2000, p. 577). Nishiyama et al. further claim that even a simple modification of processes for increased efficiency can become a burden to the organization. The article also identifies three characteristics that have been found common between different organizations in the corporation (Nishiyama et al. 2000, p. 578):

1. Parts of the problem are known, but the entire problem is not understood.
2. The methods to tackle the known problems are not clear.
3. The organizations are not confident that their methods will produce desired results.

Consequently Nishiyama et al. conclude that it is more important how well the transfer is handled than how good the chosen technology is.

SPI Success Factors. The key factors affecting the outcome of an SPI effort have understandably generated a fair amount of interest. Next, four such efforts are described briefly starting with the most rigorous effort in this group, the Dybå (2000) study, which focuses on the instrument development for validating the key factors of SPI success. This work is reported to

have followed the general recommendations for instrument development suggested in literature (Cronbach 1971; Straub 1989) that include the following steps (Dybå 2000, p. 378-379):

1. An exhaustive search of the literature for all possible key factors of SPI success and indicators for measuring them (i.e., scales); this activity covered over 600 references (Dybå 2002).
2. An exploratory study was conducted in representative companies to find possible factors and scales not suggested by literature; this activity was participated in by 54 practitioners.
3. A review of the proposed scales by experts in psychological testing and, on the other hand, in SPI to find improvement opportunities; this activity was conducted by eleven experts both from academia and industry.
4. Pilot testing of the scales in a sample similar to the target population; this pilot was participated in by twelve managers from eight companies.

Finally, the constructed measurement instrument was analyzed with actual data from 120 Norwegian software organizations. As the result of the study, Dybå reports six characteristics of SPI efforts that he claims to predict SPI success well (2000, p. 370):

- Business orientation.
- Leadership involvement.
- Employee participation.
- Concern for measurement.
- Exploitation of existing knowledge (learning by experience).
- Exploration of new knowledge (learning by experimentation).

It is interesting to notice that all these six factors are organizational ones. Dybå does not report the limitations of the conducted study but states that the validity demonstration is a continuous process where, based on Nunnally and Bernstein (1994), the *use* of the measurement instrument is validated rather than the instrument itself.

Wilson et al. (2001) provides an alternative view on SPI success factors as a result of their work towards a framework to enable companies to self-assess readiness to implement an SPI program successfully. In this study, the Jeffery and Berry (1993) metrics framework was adapted to SPI and it was concluded that there are four key perspectives that should be considered in SPI efforts: context, inputs, process, and products. Wilson et al. further adapted the Jeffery and Berry questions for the evaluation of a metrics program under these perspectives and ended up having 29 questions in their measurement instrument. This questionnaire was then completed by SPI coordinators in seven companies and in addition to this, in each company three groups of 4-6 people conducted in-depth discussions to answer the same questionnaires. The three groups represented different staff groups: developers, supervisors/team leaders, and senior managers. As the result of the study, Wilson et al. report fairly strong cumulative evidence that the successful companies were better under the four perspectives studied, and the questions that appeared to be significant indicators of the difference between successful and unsuccessful companies were the following (Wilson et al. 2001, p. 141):

- Was senior management commitment available?
- Was the SPI program staffed by highly respected people?
- Were the important initial processes to be improved defined?
- Were capabilities provided for users to explain events and phenomena associated with the program?

Wilson et al. note that since the framework was validated in only seven companies, the results are better suited as a source of hypothesis for future studies than for making conclusive decisions.

Hall et al. (2002) report another study of implementing SPI and report in it the key implementation factors that they extracted from literature:

- Human factors: SPI leaders, management commitment, and staff involvement.
- Organizational factors: communication and resources.
- Implementation factors: SPI infrastructure, setting objectives, tailoring SPI, and evaluation.

After the compilation of this list of factors from literature, Hall et al. designed a questionnaire based on it and mailed it to 1,000 software companies in the UK. They finally managed to get 200 responses to their survey but only 85 of them could be used in the analysis since not all the companies had software development functions and had attempted to conduct SPI efforts. The survey results suggest that SPI is progressing fairly well in the UK software industry and management benefit is generated by SPI even though the sample is not progressing as quickly as some of the experiences described in the literature. Key findings of the analysis include a statistically significant relationship between resources and SPI success suggesting that adequate resourcing in the companies is a problem. Other results of the analysis indicate that even if companies claim to have objectives for SPI, many of them have problems tracking and evaluating SPI adequately.

The last example of success factors is the Richardson (2002) study considering the required SPI model characteristics for small software development companies. The paper reports on an improvement model development effort which was required to demonstrate the following characteristics “reflecting financial constraints and the availability of fewer people (National Competitiveness Council 1999)” (Richardson 2002, p. 108):

- Relate to the company’s business goals.
- Focus on the most important software processes.
- Give maximum value for money.
- Propose improvements which have maximum effect in as short a time as possible.
- Provide fast return on investment.
- Be process-oriented.
- Relate to other software models.

- Be flexible and easy to use.

Matching these characteristics is further claimed to “give the effect of maximum improvement within the organization” (Richardson 2002, p. 108). Each of the above characteristics is then justified in the paper but in many cases the justifications appear as superficial as the set of characteristics and their origin.

SPI Case Studies. A study of SPI efforts reveals a number of individual SPI efforts that have similar goals as the present study. However, a closer study of the efforts tends to bring up specific issues that differentiate them after all from the present study or, on the other hand, getting enough information about the proposed approaches proves too hard to build on the reported studies. For example the Kautz et al. (2000) study reports an action research effort where a small company hired two university students to customize and implement the SPI effort based on the IDEAL SPI model (McFeeley 1996). This study appears relevant from the point of view of the present study, but noticing that a small company had to hire two people to form an internal SPI group and work on it on a full-time basis makes the approach unrealistic for most small companies, and thus the study is not considered closer here. Another example is the Isacson et al. (2001) study reporting on accelerating CMM-based improvement efforts with the Accelerator Model and method. This study reports as its justification the common conception that improvement programs are often laborious and have slow progress. It is further claimed that to address the needs of many companies working on SPI efforts the “Concepts must have a track-record of proven success in other organizations and they should be directly transferable and adoptable. This requires packaged knowledge, ready to use and including a suitable level of support” (Isacson et al. 2001, p. 23). The fundamental idea of the Accelerator approach is reported to be the systematization of the way improvements are carried out and not prescribing any improvements in itself. The following areas are further reported as the key areas of the method (Isacson et al. 2001, p. 24):

1. Commitment and communication.
2. Infrastructure and resources.
3. Improvement planning.
4. Improvement baselining.
5. Improvement execution.
6. Training.
7. Follow-up.

There are three basic problems with the presented Accelerator approach from point of view of the present study. That is, the provided information gives a very superficial picture of the approach, the article reports that the improvement efforts were supervised by external consultants as internal SPI efforts did not make progress on their own, and the experiences are reported from a large company, Ericsson. Thus the Accelerator approach does not appear to provide much support for the present study focusing on SMEs where, e.g., the term *laborious* can be expected to have quite a different meaning than in a large company reported in the study by Isacson et al.

Finally, there are two more studies that appear very promising but still have problems similar to the other attempts just described. The first one of them is the Kuvaja et al. (1999b) study describing an SPI effort tailored for small enterprises. The paper reports on a TAPISTRY approach that has goals comparable with the present study – for example, the study claims that the companies were expected to invest only two days to learn the main concepts of process assessment and improvement and apply them in the participant’s own organization (p. 154). However, attempts to find more detailed information about the approach failed, and no other information could be found on the approach but this fairly short paper reporting on the method workshops. Thus further study of the approach was not possible.

The paper by Calvo-Manzano Villalón et al. (2002) describes experiences of applying SPI in SMEs with the MESOPYME method. The goals of the MESOPYME method are very similar to the present research effort since it is targeted for SMEs where the financial, schedule, and resource constraints are reported to cause difficulty in using the current SPI methods like ISO 15504 (ISO/IEC TR 15504-1 1998). The MESOPYME method approaches SPI with a concept called Action Package, and the results of applying the requirements engineering Action Package in three organizations are reported in this paper. The MESOPYME method is reported to have four stages whose objectives are similar to the ones of the IDEAL model (McFeeley 1996). The Action Package objective is reported to be “fast technology transfer, and to implement a process in 5 to 7 months, with a little investment of money and person” (Calvo-Manzano Villalón et al. 2002, p. 265). However, the implementation of the method is reported to start with the establishment of an organization consisting of three groups: a software engineering process group, a software change control board, and a working group to define and develop new processes. The two first groups are reported to be permanent ones in an organization while the last one with four people has the responsibility of defining and piloting new processes; after this the working group is broken down and the software engineering process group becomes the process owner. As the average effort spent in the improvement solution activities with the RE Action Package, Calvo-Manzano Villalón et al. report 3.87 man/month external effort and 7.5 man/month internal effort. The study is reported to have been conducted in three companies with 10-90 people in software development. However, as a whole the data on organizations, projects, used resources and implemented actions are reported in such a superficial manner that assessing this study is very difficult. Other information on the MESOPYME method in English include a journal paper (Calvo-Manzano Villalón et al. 1997) and the doctoral thesis of Professor Calvo-Manzano Villalón (personal communication, December 18th, 2003). Unfortunately, these publications provide little additional information to the article reported above, which makes building on the experiences from this work hard.

3.2.7. Summary

A review of the state-of-the-art in RE research revealed that little attention has been paid to the application domain, methods, integrated frameworks, and introduction level RE in the past. Unfortunately, these areas are also seldom suggested as promising future research topics. In this section, some of the more interesting research efforts were considered more closely only to find out that their actual focus was much too advanced for the purposes of the present study.

The agile software development approaches provide more concrete support to introduce some order in the often chaotic software development practices in small organizations. However, the fundamental goals of emphasizing people and interaction over documented requirements and

practices have lead to situations where documenting activities are deferred for as long as possible and preferably implemented in a follow-up project. This approach clearly contradicts the basic systematic RE which is by and large built on documenting and analyzing requirements. Thus even the emphasis on close customer interaction cannot prevent from concluding that current agile approaches provide little tangible help for RE.

The situational method engineering suffers from a very different kind of problem than the agile approaches since it suggests that actual methods should be constructed only after project specific needs are known. Namely, the present work tries to find an easy to adopt solution for doing RE, so a method construction task before moving to requirements development clearly contradicts this goal.

In the present work, domain engineering was used to refer to software development approaches that are based on a priori experience or knowledge about the task and domain at hand. Domain engineering focuses on the reuse of past experience in a particular domain in the form of different kinds of assets, such as documents and patterns. The theoretical justification for this approach was found in the research stream on cognitive fit suggesting that methods should be studied from the viewpoint of their benefit to particular classes of applications in order to be able to match method to application task.

The technology transfer literature was also considered from two viewpoints. First, the problem of introducing new technology in practice was explained. This problem was succinctly summarized in one study claiming that improved methods of technology transfer limit or enable the adoption of useful new software engineering technologies and practices. Second, the research on factors affecting method adoption in industry was exemplified by two studies on the topic. The first one of these summarized the seminal studies on the topic and reported that in the conducted study method adoption intentions were driven by usefulness, voluntariness, compatibility, and subjective norm. The second study reported comparable results even if they were not identical. Overall, these two studies seem to present the state-of-the-art in this area and studying the characteristics of an innovation in technology transfer – of diffusion – studies appears to be a justified decision.

The last section on process improvement literature included a short introduction to business process reengineering but focused on software process improvement issues. The main difference between these approaches is that business process reengineering in general represents a radical approach to change while software process improvement is usually suggested to be a long term and iterative process. A number of software process improvement articles reporting on related problems, success factors, and case studies were then summarized to provide a look at typical findings from the studies in each category.

3.3. Technical Issues

The conducted literature surveys brought up technical issues that appeared important for the present study. Thus in this section three studies on central elements of methods are summarized, and the central findings of a study on RE tool and technique selection are reported. The section is closed with a summary.

Before going into the technical issues it should be noted that organizational issues (Table 13) have been claimed to play a substantial or even primary role in systems failures (Doherty and King 1998). The importance of these issues in software development was apparent also in the review of the SPI success factors presented in Section 3.2.6.

However, a technology driven approach has also been reported important in systems development (Hornby et al. 1992, p. 165):

Systems analysts do not claim to have knowledge of organizational issues in IT systems, and there is no evidence that they are encouraged or rewarded for considering such issues. In fact it could be said that the reward and control systems within which analysts work actively encourage them not to consider them. They are rewarded in the main, for delivering technically sound systems on time and to budget.

In the present study, organizational aspects are excluded from further examination. This approach is based on the finding in the practical study quoted above and the study by Beecham et al. (2003) that companies with low maturity are not in a position to concern themselves primarily with organizational issues as they tend to have more urgent technical and project problems (p. 61).

3.3.1. Central Elements of Methods

Avison and Fitzgerald (2003, p. 31-32) claim that an information systems method should possess five attributes. These attributes are a series of phases depicting a sequential process, a series of techniques to evaluate costs and benefits of different solutions and, on the other hand, to formulate detailed design for the system, a series of tools for systems analysis, a training

Table 13. Organizational issues in system development (adapted from Doherty and King 1998, p. 45)

Major categories for organizational issues	Detailed issues
Organizational alignment	Organization’s culture Organization’s structure Distribution of power
Organizational contribution	Cost-benefit analysis Information systems strategies Prioritization of work to organizational requirements Consideration of organization’s future needs
Human issues	Training Job re-design Consideration of health and safety/ergonomics issues Management and user education User motivation
Transitional issues	Timing of the implementation Perceived level of organizational disruption
Systems integration	Interfaces to existing systems

scheme, and a philosophy which is concerned with the basic assumptions and viewpoints of the method.

Ter Hofstede and van der Weide (1992) and Andersen et al. (1990) describe respective frameworks outlining comparable method components. The framework by ter Hofstede and van der Weide (Figure 11) addresses the same aspects of a method as Avison and Fitzgerald (2003) in the following way: a way of thinking compares with a philosophy, a way of control compares with series of techniques and phases that are, e.g., used to evaluate the costs and produce plans, a way of modeling compares with a series of techniques used to model the product, a way of working compares with a series of phases, and finally a way of support compares with a series of tools and training. There are, of course, some differences in the scope of the different components but at a conceptual level the similarities between the approaches are evident. Andersen et al. (1990), on the other hand, provide a somewhat simpler characterization of methods by proposing that a method should outline its intended application area, perspective (i.e., philosophy), and guidelines. The guidelines are further divided into techniques (covering both product and process aspects of development), tools, and principles of organization (the project management point of view).

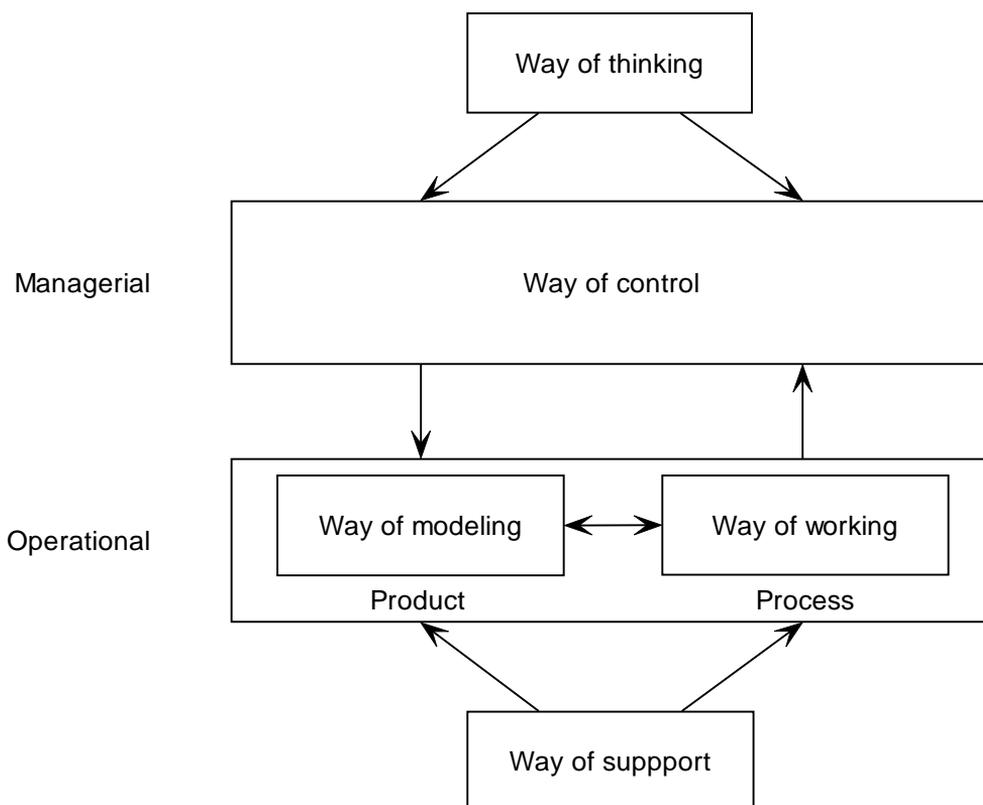


Figure 11. Framework for methods (ter Hofstede and van der Weide 1992, p. 57)

3.3.2. Key Issues in RE Tool and Technique Selection

The study by McPhee and Eberlein (2002) was conducted as an online survey to gain insight into how software developers view the RE phase and what kind of tools and techniques they consider as being most useful for it. Another goal for the study was to clarify the possible differences between time-to-market (TTM) and non-TTM projects in their approaches to RE. Approximately half of the 25 survey respondents had 10 years or more of software development experience. The following review focuses on the results of the survey as related to RE technique selection and ends with a summary of the conclusions.

One part of the McPhee and Eberlein study addresses the level of knowledge and perceived usefulness of different RE tools and techniques. In the survey the tools and techniques were provided as a list where the respondents were to indicate their level of knowledge on a scale of 1 to 5. If a respondent was in at least some way familiar with a technique, the usefulness of the technique both for TTM and Non-TTM projects was also to be rated. Table 14 shows the top ten results of this part of the study – first the ten most familiar techniques and then the ten most useful techniques for TTM and Non-TTM projects. Even though the techniques seem to vary quite a bit, there are some interesting points to take into account:

- Scenarios/Use Cases, Semi-Formal Modeling, and Change Management all fitted within the top five techniques in all three categories.
- Requirements Prioritization was the most useful technique for TTM projects but did not fit within the ten most useful techniques for Non-TTM projects.
- Requirements Tracing, Viewpoint-Oriented Techniques, and Designer as Apprentice were considered useful for Non-TTM projects, but not as useful for TTM projects.

Table 14. Top 10 RE techniques (McPhee and Eberlein 2002, p. 20)

Rank	Familiarity	Usefulness (TTM)	Usefulness (Non-TTM)
1	Scenarios/Use Cases	Requirements Prioritization	Requirements Change Management
2	Semi-Formal Modeling	Requirements Change Management	Semi-Formal Modeling
3	Informal Modeling	Scenarios/Use Cases	Requirements Reviews
4	Requirements Change Management	Semi-Formal Modeling	Scenarios/Use Cases
5	Evolutionary Prototyping	Requirements Testing	Requirements Checklists
6	Interviews	Evolutionary Prototyping	Requirements Testing
7	Requirements Prioritization	Requirements Reuse	Requirements Tracing
8	Requirements Reviews	Interviews	Viewpoint-Oriented Techniques
9	Throw-away Prototyping	Requirements Reviews	Interviews
10	Requirements Checklists	Requirements Checklists	Designer as Apprentice

The reasons for choosing RE techniques and their desirable qualities were also studied in the survey (Table 15 and Table 16). The facilitation of good communication is the second most important item in both the lists, so the importance of good communication is evident. The most important quality for the techniques was the ease of use which matched quite well the most important reason for choosing a specific technique – having previous experience of it. The four most attractive aspects of RE were defining common goals/scope, which was identified by 60% of respondents, gaining domain knowledge (28%), exploring possibilities (24%), and team building (8%). To provide a perspective to attractiveness, also the most unattractive aspects of RE were listed. The results revealed that difficulties in communicating with stakeholders was the most disliked attribute of RE (identified by 48% of the respondents), followed by documentation (28%) and changing requirements/feature creep (24%).

Table 15. Reasons for choosing RE techniques (McPhee and Eberlein 2002, p. 22)

Reasons for choice of technique	Percentage of respondents who identified the reason
Previous experience with technique	28%
Facilitates good communication	24%
Company standard	16%
Tool support available	8%

Table 16. Desirable qualities of RE techniques (McPhee and Eberlein 2002, p. 22)

Desirable quality of RE technique	Percentage of respondents who identified the quality
Easy to use	32%
Facilitates good communication	28%
Facilitates capture of complete set of requirements	24%
Allows for or incorporates traceability	20%
Improves the product quality	8%
Tool support	8%
Incorporates prioritization	4%

The conclusions of the study are the following (McPhee and Eberlein 2002, p. 23):

- There exists a general lack of knowledge regarding available RE techniques.
- RE technique usefulness is largely determined by familiarity.
- Individuals who worked primarily on schedule-driven projects generally had less familiarity with RE techniques than individuals who worked primarily on budget or scope/quality-driven projects.
- Good communication was a key factor in successful RE.
- It was important that requirements be unambiguous.

- The perceived usefulness of RE techniques for TTM projects correlated with the perceived usefulness for Non-TTM projects.
- RE techniques must be easy to use.

Overall, the findings of McPhee and Eberlein are not surprising but appear comparable with other conducted studies. However, the reliability of the study findings is not self-evident as the article does not report how the study was actually conducted and its limitations are not reported even if numerous details suggest that there may be some limitations. For example, the sample size is fairly small; nothing is said about a possible pilot study; and the time used to answer the questionnaire is not reported.

3.3.3. Summary

In this section, different frameworks characterizing information systems development methods were summarized. Even if the proposed frameworks used different terminologies, common concepts across them were evident. Thus it appears evident that using such a framework would prove beneficial both in the method development phase as well as in the method learning phase.

Next, the findings of a study on RE tool and technique selection were summarized. The most important findings include the importance of easy to use solutions; lack of knowledge in methods, techniques, and their use; and a claim that the usefulness of an RE technique is largely determined by its familiarity. Even if this study has some obvious methodological shortcomings, the results are supported by anecdotal evidence in the field. For example, a study on an RE improvement effort reports the ease of learning and use to be crucial for new methods and models (Kauppinen et al. 2002, p. 49).

4. PROBLEM ANALYSIS

In this chapter, a problem analysis for the planned process improvement effort is presented. First a state-of-the-practice investigation in industry is presented in Section 4.1, and then an organizational context (Section 4.2) and an application domain for the present study (Section 4.3). Section 4.4 closes the chapter with a summary of the problem analysis findings.

4.1. State-of-the-Practice Investigation in RE

The current state-of-the-practice should be the starting point for improvement efforts for both researchers and practitioners (Davis and Hickey 2002, p. 107). In addition to personal experiences, surveys on the topic can provide a wider perspective on the situation. Unfortunately, the published state-of-the-practice surveys in RE tend to focus on specific research questions and do not really provide a comprehensive picture of the situation in general (see e.g. Curtis et al. 1988; Lubars et al. 1993; El Emam and Madhavji 1995; Nidumolu 1996; Chatzoglou 1997a; Guinan et al. 1998; Kamsties et al. 1998; Hofmann and Lehner 2001). Thus a new state-of-the-practice investigation was conducted in twelve companies with the aim of forming a general view of the RE state-of-the-practice in small and medium software houses in Finland (Nikula et al. 2000a; Nikula et al. 2000b; Nikula et al. 2000c). After this study, at least one further state-of-the-practice study has been conducted in RE and reported in two articles (Laplante et al. 2002; Neill and Laplante 2003). This section summarizes the results of the conducted investigation and compares them with other research results where possible.

4.1.1. Research Method

The investigation was conducted as structured interviews covering RE practices, the company background, working methods in general, and knowledge in RE. The questions were compiled into a four-page questionnaire that was used to guide the interviews. Most of the questions were multiple choice ones or could be answered with only a few words, but some open questions were also included to provide the interviewees with the possibility of expressing their own viewpoints. The interviews were conducted with key informants of the companies consisting of ten highest level people in their area (development or quality directors, chief executive officers, and a chairman of the board), six second highest level (quality or information technology managers), and one researcher responsible for software development. In seven cases specialists were also present so that altogether 32 people participated in the interviews. All the interviews were conducted on company premises by the author of this thesis and lasted between one and five hours with an average of about 1.5 hours; the interviews were audio recorded and in some cases further clarification was sought with email and telephone contacts.

4.1.2. Background

The investigation was conducted as semi-structured interviews in twelve companies located in six different cities. All but one company competed in international markets with world-class products, and three companies had international development teams. The number of employees varied from four to over one thousand with a median of 25 persons. Most of the companies developed general business information systems but also telecommunications, digital media, data security, industrial applications, and software engineering tools were produced by them. The project types covered custom software and component-based projects as well as

commercial product development. A typical effort for a project with a project plan was between 12 and 36 person months, and typical project duration was between 6 and 12 months. The product sizes were estimated in code lines and a majority of them were between 10 and 60 thousand lines. Although the literature contains a number of other state-of-the-practice surveys in RE, there is only one other study that has focused on small and medium enterprises, namely the study of Kamsties et al. (1998).

4.1.3. General Level of Practices and Knowledge

To get a general idea about the companies' approaches to software engineering, they were asked about their process maturity and specialization in roles and tools. As a reference point for maturity, the Software Engineering Institute's capability maturity model (CMM) was used (Software Engineering Institute 1995). Most of the companies did not recognize CMM, and none of them had had their capability assessed according to CMM by an outside party. However, three companies had done their own evaluations, where one company reached level 3 and two level 1. Otherwise one company had an audited quality system, nine had accepted their processes internally, and two had informal internal processes.

The specialization of the companies was studied by asking what specialized roles their software developers had and what special purpose tools they were using for software development. The most common specialist role for team members was designer of the user interface, database, or alike, which was present in seven companies (*Des. UI/DB* in Figure 12). The next most common specialist groups were *systems analysts* and *technical writers*, both were found in six companies. Three companies had *testers* and five did not have any such specialists, but called all employees *developers*. It was also noteworthy that only half of the systems analysts worked only in the RE area while the other half worked also as project managers or developers.

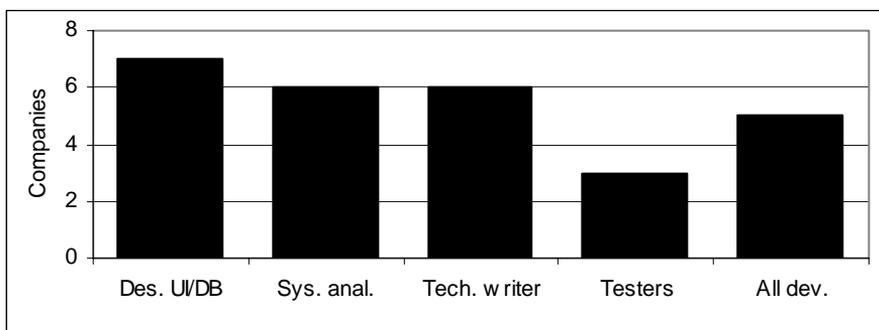


Figure 12. Team member roles in the companies

On the tool side the most commonly used tool was a *configuration management tool*, which was used by ten companies (Figure 13). Eight companies used some special *testing tools* and four had *CASE tools* in use. No company had *requirement management tools* in use. The absence of specialized tools has also been noted in other studies. For example Lubars et al. (1993) report that a minority of the projects they studied used or had used upper-CASE tools. In a study by Hofmann and Lehner (2001) some RE teams experimented with commercial RM tools but in all but one case the tools interfered with rather than supported RE activities.

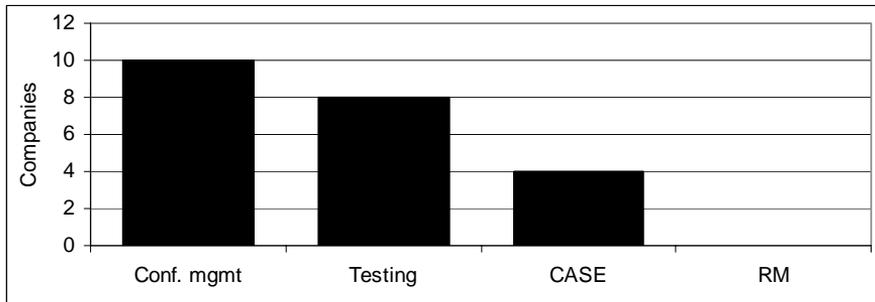


Figure 13. Use of tools in the companies

Since it was suspected that there would be shallow knowledge in RE and RM tools in the companies, two questions addressed these areas. There were two lists from which the interviewees were asked to indicate familiar titles. The book list consisted of seven references (Gause and Weinberg 1989; Davis 1993; Jackson 1995; Sommerville and Sawyer 1997; Thayer and Dorfman 1997; Kovitz 1998; Robertson and Robertson 1999), and the tool list of the 13 RM tools in the INCOSE survey at the time of the investigation in 1999 (INCOSE 2003). The question about these books and tools was whether the name was known; reading of the books or use of the tools was not required. Four interviewees recognized one book and one person two books, while ten interviewees did not recognize any book. From the tool list, seven interviewees recognized from 1 to 3 tools and eight did not recognize any of them.

4.1.4. Requirements Engineering Practices

RE practices in the studied companies varied quite a lot, but in most cases they seemed to be common sense practices based on general knowledge. Most companies used word processor or spreadsheet programs to manage their requirements, but single database, change management system, and email solutions were also used. The requirements were usually represented in natural language and in some cases with semi formal methods (e.g. unified modeling language, UML), while formal methods were not used in any company. Four companies had an RE process defined and another four had a requirements document structure defined, but only two companies had them both defined. For the requirements document some further details were considered, and in the cases a structure was defined, it was also clearly thought out. As a rough estimate of a typical requirements count seven companies estimated tens, four companies hundreds, and one company thousands of requirements.

The RE practices in the companies were assessed using the REAIMS Top Ten guidelines (Figure 14). The most adopted guideline in the companies was a standard document structure, which in general meant a document template. However, even this guideline was applied in a standard manner only in four of the companies and two companies used templates on a normal basis.

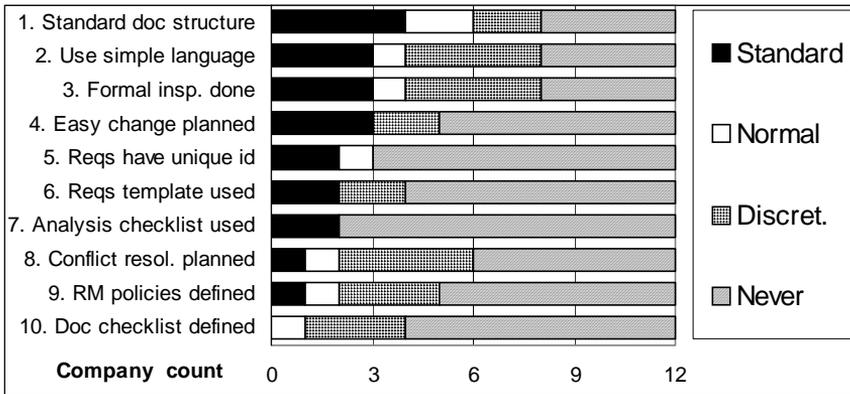


Figure 14. REAIMS Top Ten questions and ratings in Pareto order

Figure 15 shows the REAIMS Top Ten point gains for the companies. This result reflects the overall feeling of the situation in these companies quite well – some companies had clearly invested in RE, which is shown as a point gain from 15 to 28 of the maximum 30 points. Most of the companies were still in the initial stages of their RE practices.

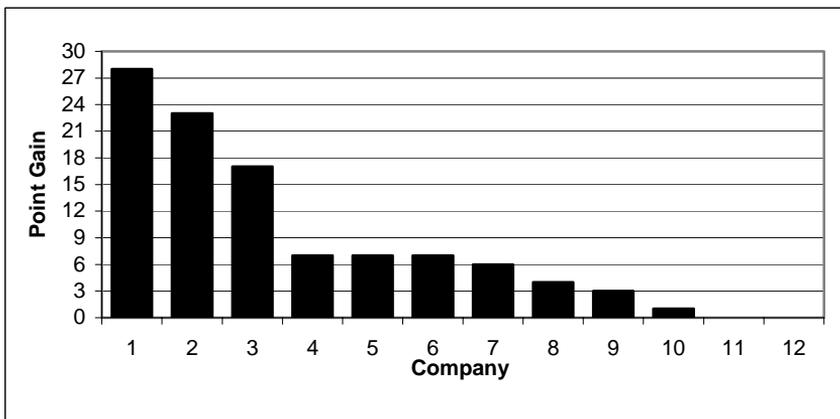


Figure 15. REAIMS Top Ten -point gains for the companies. The maximum point gain was 30; the best result was 28 points and two companies did not get any points

The latest state-of-the-practice study in RE reports on the use of RE practices in industry (Laplante et al. 2002; Neill and Laplante 2003). In this web-based survey, 194 responses were received from 1,519 people invited to answer the questionnaire. The most interesting findings of this study include the use of different requirements notations: 51% of the respondents are reported to have used informal notations, 27% semiformal notations, 7% formal notations, and 15% of the respondents are reported to have used other techniques (Laplante et al. 2002) or not to have responded to the question (Neill and Laplante 2003). As more detailed information, it is reported that over 50% of the respondents used scenarios or use cases in the requirements phase, 33% indicated that they did not use any methodology for requirements analysis and

modeling, and finally, Laplante et al. (2002) report that 32% of the respondents indicated that they worked on projects requiring 25 or less pages of requirements and 50% reported to have worked on projects with less than 75 requirements.

4.1.5. Development Needs and Attitudes to RE Improvement

The study included a section for development needs and attitudes to RE improvement in general. Ten of the twelve companies found RE to be essential for their business success. Five companies also considered their RE process improvement activities to have high importance, while six considered them to be of medium importance, and one rated them as a low importance activity. The specific development needs are shown in Figure 16 focusing on the needs that were common to at least nine of the twelve companies. These development needs cover requirements development, management, and documenting issues with tool support for RM, suggesting that support is needed in all areas rather than in some specific area.

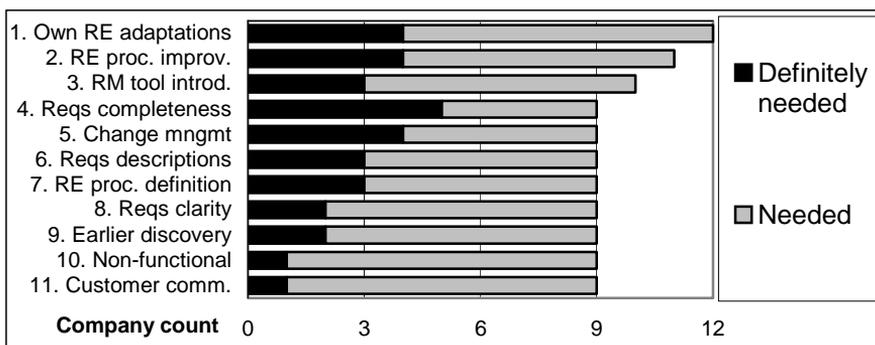


Figure 16. Development needs in RE in Pareto order

Another study addressing the development needs is described in Kamsties et al. (1998). This study approaches the needs in quite a different way, since the participants were asked to put six RE topics both in order of relevance in their environment in general and in order of priority within their own improvement plans. The topic of highest relevance was requirements modeling, followed by requirements document improvement. In the priority aspect, the requirements document shared the second place with inspections and tools while requirements modeling again took the first place.

Considering the time used for RE, an interesting finding in the investigation was that only one company (8%) had a definite need to reduce the time and effort on RE. One third of the companies (33%) even thought that the time or effort for RE could and should be increased. This topic was addressed also in the latest state-of-the-practice study (Laplante et al. 2002; Neill and Laplante 2003) in which it was found that 52% of the respondents did not consider the time their companies used for RE sufficient and only 29% percent felt that the requirements effort was sufficient in their companies.

4.1.6. Expectations for Academia

The last part of the conducted state-of-the-practice investigation clarified what the companies expected from academic research and what kind of research approach would benefit them the most. For the companies that participated in the study, the most important goals for academic RE research were clearly education and technology transfer. The other options of a new RE method and technique development or a new tool development were perceived as much less important, as can be seen in Figure 17.

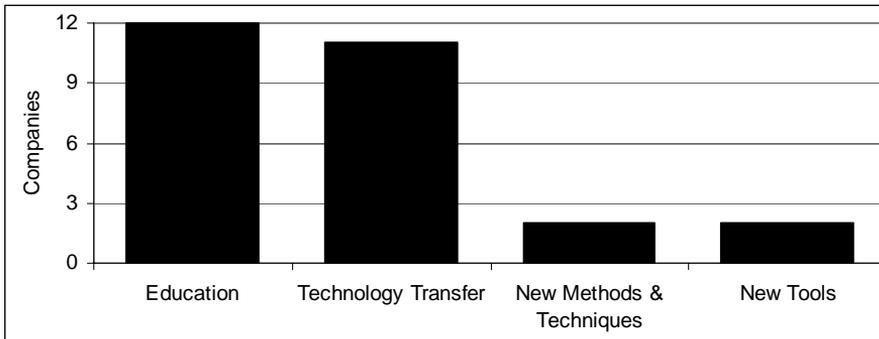


Figure 17. Expectations for academic RE research

Technology transfer had sub-options, and nine of the ten companies that answered them considered training for practitioners to suit their purposes best. Eight companies expected templates and examples to work best, while six companies wanted consultation. Two would have liked to see new technology transfer methods to be created and used, as can be seen in Figure 18.

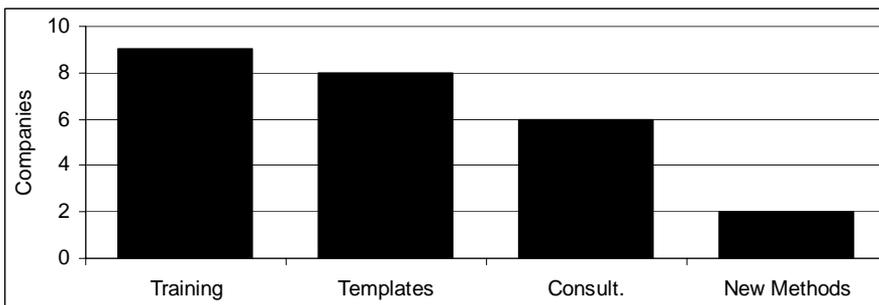


Figure 18. Preferred ways of technology transfer

4.1.7. Discussion and Conclusion of the Study

The presented state-of-the-practice study has several limitations. The number of studied companies is small and their selection was based on the availability for the investigation. The companies and their approaches to business differed greatly which makes their comparability

with each other questionable. However, the company profile of the study represents in many respects the general company profiles in Finland fairly well. For example, the study by Maansaari and Iivari (1999) on CASE tool evolution in Finland reports that 45% of the software projects in the study consisted of 3 to 5 people. It is important to notice that in this study 50% or 22 of the companies had over 500 employees which makes comparison of the study with the presented investigation otherwise questionable. The conduct of the investigation appears also to have some deficiencies. For example, it is somewhat surprising that all the companies reported a need for their own RE adaptations. Due to the limitations of the investigation, it is not taken as the only basis for further work but it is supplemented with a literature survey to identify actual problems in industry related to the observed low level of RE practices. The mentioned limitations make it interesting to notice that this study is a common reference in later research reports (e.g., Juristo et al. 2002; McPhee and Eberlein 2002; Neill and Laplante 2003).

An overall observation in the investigation of Nikula et al. (2000a; 2000b; 2000c) is that the companies that participated did not exhibit clear leadership in RE practices. Most companies lacked a strategy to start improving their RE practices, while some companies had already established basic practices. With the three most advanced companies a more thorough assessment would have surely provided more information about the practices, but for the other companies the REAIMS Top Ten assessment provided quick feedback on the general level of the practices. A fundamental problem with the studied companies was the lack of specialized people.

The results also suggest that a practically oriented RE improvement effort targeted for companies typical in the investigation should take into account the same basic factors. A typical project tends to run from six to twelve months and involves only a few people. The maturity in software engineering and RE practices is low; knowledge in the RE field is limited, use of specialized tools is limited, and specialization is rare. Some companies do have specific needs but in general help is needed in a holistic manner. Companies would like to hire people educated in the RE field, but also technology transfer is expected; the preferred forms of technology transfer are training, templates, and consultation.

4.2. Organizational Context for the Study

The conducted state-of-the-practice investigation concluded that many SMEs have low maturity in both RE and software engineering and thus they could benefit from support in the process improvement area. In this section, this organizational setting is studied based on literature on the respective topics. First, findings on SMEs, small projects, and low maturity organizations are presented, and then typical characteristics of these kinds of organizations – lack of resources and uncertainty about payback – are considered more closely. From the typical characteristics only one form of resource, lack of expertise, was anticipated at the time of the above presented state-of-the-practice investigation and this characteristic was found to be a typical one in the studied companies. The other characteristics were not addressed in the study and, consequently, no practical insights were gained from them.

4.2.1. SMEs and Small Projects with Low Maturity

Focusing the present study on small projects and organizations is supported not only by the conducted state-of-the-practice study but also the general importance of these areas. The

European Union (EU) (The European Commission 2003) reports that 99% of all enterprises in the EU are SMEs that provide a job to about 65 million people. This is consistent with the estimates in the USA (The White House 2001, p. 9): “Small businesses represent 99 percent of businesses, employ more than half of the American work force, and create two-thirds of the net new jobs.” It should be noted that the EU SME definition (p. 27) is different from the small business definition in the USA as, for example, the maximum number of employees in the USA varies between different industries being often 500 people (United States Small Business Administration 2003). More concrete figures are estimated by Fayad et al. (2000, p. 117) who estimate, based on the US Census Bureau’s 1995 County Business Patterns report, that 99 percent of the US data processing companies had less than 250 employees providing work to over 770,000 people. In the same vein, the question of the importance of small projects is touched upon in software engineering literature. For example, Humphrey (1998, p. 201) reports an observation that engineers invariably write large programs as collections of small ones and if best methods are not used for writing small programs, it is unlikely that they are used when writing large programs. In their column Laitinen et al. (2000, p. 106) state also that there is evidence of the higher per-capita productivity of small groups in comparison with large ones.

Small businesses have been claimed different from large ones in numerous articles. Thong et al. (1996) state that small businesses are unique from the IT perspective and Thong (1999, p. 188) summarizes the related literature comprehensively with an overall estimate that the studies in IS adoption in large businesses are unlikely to be generalizable to small businesses. One reason for this is that the skills, time, and staff that are necessary for planning are a major issue in small businesses whereas in large businesses they are not. Consequently, it is suspected that organizational theories and practices applicable to a large business may not fit a small one. It should also be kept in mind that results from the IS adoption domain are not necessarily applicable in the context of the present work even though Riemenschneider et al. (2002) report that the models used in the tool adoption context appear to suit also the method adoption context (p. 58).

Laitinen et al. (2000) discuss the problems of scalability and claim that common recommendation for adapting methods to small project use are not really useful. The common ideas of “just scale it down” and “developers in small groups must play many roles” are good advice on paper but experience problems when tried in practice:

- If much of the software process is devoted to managing multiple groups rather than dealing with some unique characteristic of software, then identifying what to keep and what to drop is not obvious. If you don’t drop any of the methods, then how, exactly, do you go about scaling them down? (Laitinen et al. 2000, p. 105)
- Clearly, members of a small development group play many roles, but these are not necessarily the same ones needed for a larger organization. The difficulty instead, is to decide what roles are important for developing good software and what roles are important for coordinating and managing multiple groups. (Laitinen et al. 2000, p. 106)

Laitinen et al. continue by claiming that having irreplaceable people in large groups is a weakness while in small groups it is a necessity. The discussion on method scalability is closed with an observation that it is unclear whether many methods can be scaled without fundamental

changes in them. The same authors discuss software process improvement in the small in another column (Ward et al. 2001) and conclude that processes “done right” for small companies are probably less than ideal from the critics and implementers point of view since the central characteristics of such processes are simplicity and robustness instead of the more common “best practices.” This “best practice” SPI approach is also challenged by the doctoral study of Dybå that is summarized in Dybå (2002). In this article Dybå states as one of the conclusions of his study that for software organizations to be able to enable SPI “a different context than that proclaimed by current *best practice* SPI approaches must be created” (2002, p. 389). Another interesting finding in this study is that large successful and small successful organizations had fundamentally different approaches to SPI. It was found that “small software organizations, in turbulent environments, require learning strategies that are closely aligned with explorative behavior, while at the same time promoting the exploitation of past experience” (Dybå 2002, p. 389).

The Beecham et al. (2003) study (p. 61), on the other hand, researched SPI problems in twelve companies and managed to establish a relationship between the company’s maturity level and characteristic problems. Even though the small sample size does not provide for conclusive results, the study results suggest that “low maturity companies suffer from project and technical problems while high maturity companies are more burdened with organizational problems” (Beecham et al. 2003, p. 24).

The surveyed literature provides mainly anecdotal evidence on the differences between small and large organizations and, on the other hand, between low and high maturity companies. Still, literature findings suggest that these organizational factors are best studied separately (Thong 1999, p. 188; Fayad et al. 2000, p. 118). Examples supporting this approach include a finding that the skills, time, and staff for planning are a major issue for small organizations while for large ones they are not and, on the other hand, an observation that large successful and small successful organizations have had fundamentally different kinds of approaches to SPI. The slowly increasing number of articles on SPI efforts in SMEs reviewed in Section 3.2.6 also reflects this situation.

4.2.2. Lack of Resources

A lack of resources for process improvement is mentioned in many articles. A closer look at these articles reveals that the term *resource* is often used to refer to three different aspects of resources: the quantity, availability, and quality. The quantity of resources refers in general to the financial resources (money) and the possibility to pay for the investments associated with process improvement efforts while the availability refers to the time companies are willing and able to allocate to process improvement efforts. The quality of resources refers to the expertise companies have in process improvement.

Quantity and availability of resources. In the studied literature, the quantity and availability of resources are often reported together so these two aspects are not separated here, either. For example, Fayad et al. (2000, p. 117) state that resources are severely limited in small companies and Calvo-Manzano Villalón et al. (2002, p. 262) report that their approach was developed to solve the problems related to high costs and extensive time consumption often related to SPI efforts. The lack of resources appears to be a potential problem in large companies. This is also exemplified by three studies: the study by Nishiyama et al. (2000, p. 577) claims that even a

simple modification of processes for increased efficiency can become a burden to the organization, the study by Fowler et al. (1998, p. 144) reports that the users of their pilot website reported very limited availability of internal resources to support RE introduction efforts, and finally the study by Lycett et al. (2003, p. 82) found that generic processes need tailoring which requires substantial up-front efforts and it was suspected that this could explain why many reuse efforts yield neither use nor reuse. The lack of funds as such has been reported as a problem in small companies both in IS adoption studies (Thong et al. 1996) and software process improvement studies (Kuvaja et al. 1994; Kilpi 1997) while the problem of finding time for SPI efforts has been reported as pressure from the competitive environment (Kaindl et al. 2002, p. 117) and as an importance of saving time in acquiring information (Fowler et al. 1998, p. 144). Also the availability of people for SPI efforts has been claimed a problem for small companies (Richardson 2002, p. 110). Overall, the lack of money and time appears a common problem in industry and especially in small companies.

Quality of resources. The quality of resources refers in general to the expertise companies have in the process improvement. Having expertise in a specific topic is beneficial for two reasons: first, an expert can identify improvement opportunities, and second, an expert knows how to take advantage of those opportunities. In this section, literature on different aspects of expertise is summarized: lack of expertise in many companies, failure to benefit from available technologies, need for learning, use of external specialists in many SPI efforts, and some demotivators for SPI efforts. In this section, a clear distinction is not made between different company sizes since the problems are somewhat common to companies of all sizes with one exception – it has been claimed that small companies cannot afford to maintain substantial SPI expertise within their companies (Kuvaja et al. 1999a, p. 150).

The lack of expertise in companies is attested by numerous articles. Nishiyama et al. (2000) report that it was possible to identify three common characteristics between different organizations in their corporation; two of these characteristics were “Parts of the problem are known, but the entire problem is not understood” and “The methods to tackle the known problems are not clear” (p. 62). This situation appears similar to the Harkness et al. (1996, p. 353) study where it was reported that “we knew we had to fix problems, but we did not know how,” and the Fowler et al. (1998) study where the people accessing RE related information from a website were new to RE as well as software engineering in general – this result was obtained both by a user questionnaire and website usage information. One reason for this situation is provided by Nishiyama et al. (2000, p. 577) who report that developers and organizations “cannot readily afford the time or money needed to learn and incorporate new techniques and technologies in their development practices.” Kaindl et al. (2002, p. 117) report also that in many cases “there is little or no opportunity for technical consultation, leaving the software engineer to feel lost and isolated.” In the Chau and Tam (1997) study on open systems adoption, the perceived barriers to adoption were found to be significant; Chau and Tam refer to this problem as technology adoption as a knowledge barrier, as originally suggested by Attewell (1992). This situation is actually not surprising as Blili and Raymond (1993) report that SMEs tend to employ generalists rather than specialists. Concluding that few companies appear to have expertise in SPI, it is no wonder that they have problems in figuring out how to start improvement and what experts to use (Kuvaja et al. 1999a, p. 150).

The need to develop expertise, or at least learn more, is apparent when technology transfer and process improvement initiatives are launched. For example, in the study by Morris et al. (1998), the highest priority problem area was training, and training is also one of the CMM level 3 key

process areas (Software Engineering Institute 1995). There are a number of studies suggesting that a culture for continuous process improvement and learning has to be established to achieve real changes (e.g. Caron et al. 1994, p. 247; Harkness et al. 1996, p. 353; Sweeney and Bustard 1997, p. 272). The importance of this change is described well by Sweeney and Bustard (1997, p. 266) who compare SPI to learning swimming: “By analogy with swimming, this is perhaps like having a tailored training scheme rather than jumping in at the deep end!” The training issue is a research area of its own and in the present study it is not addressed in detail. It is worthwhile to note that Attewell (1992) has re-conceptualized technology diffusion in terms of organizational learning, skill development, and knowledge barriers. He states that for technology to be taken in use, both individuals and organizations must learn new skills, knowledge, routines, practices, and beliefs. A central problem is that there is a “need for learning and skill formation *in situ*” (Attewell 1992, p. 6) since knowledge cannot be transferred from the originator to the user of the technology.

It is no surprise that most of the SPI efforts reported in literature have involved external specialists, quite often one or more of the authors reporting the study. For example, the Kautz et al. (2000) study was initiated when the company management contacted a university to get help in the SPI efforts, the Richardson (2002) study was conducted as an action research effort and the author was involved in the writing process together with the software development group, in the Kuvaja et al. (1999a) study the workshops were arranged as a part of a research program, and finally, in the Calvo-Manzano Villalón et al. (2002, p. 270) study it is reported that each action package development required an external effort of “3.87 man/month.”

A lack of resources has both direct and indirect consequences. Namely, SPI motivators and de-motivators are claimed to have substantial influence on the performance and the quality of conducted improvement efforts, and resources are reported as important factors in such studies. For example, the study by Kaltio and Kinnula (2000) reports that skills, motivation, and time were the most important motivators in deploying defined software processes. Baddoo and Hall (2002, p. 92), on the other hand, report that the three most important common motivators to software developers, project managers, and senior managers were process ownership, evidence, and resources. In their related study on de-motivators, Baddoo and Hall (2003, p. 31) report as common de-motivators were related to resources, commercial factors, the actual process constraints, implementation issues, and personnel factors. Furthermore, it is reported that all these three groups reported the lack of appropriate SPI skills in their companies as de-motivating: both developers and senior managers reported the project managers’ lack of SPI skills de-motivating while project managers “see the problem of shortage in SPI skills as an indirect result of high staff turnover” (Baddoo and Hall 2003, p. 31).

In short, three common resource related problems were identified in literature: lack of money, time, and expertise. These problems have been reported in very different kinds of organizations, so an SPI effort can be expected to encounter these obstacles in companies of any size. However, it can be assumed that small organizations have in general fewer resources than large ones and, consequently, the resource issue requires extra attention in small organizations.

4.2.3. Uncertainty about Payback

There seems to be a fair amount of skepticism about how well the investments in SPI efforts pay themselves back. Many base their suspicions on their personal experiences from the past

while others are unwilling to commit to long term investments amidst their daily struggle with financial issues. In this section, this topic is studied based on literature discussing the problems of small companies in particular and supplemented with problems reported in large companies. The overall payback problem comprises of three separate factors – the amount of investment, the amount of payback, and the payback period.

There is an increasing body of literature stating the need to keep the up-front costs low in SPI efforts. For example, Laitinen et al. (2000, p. 107) discuss scalability in their column and summarize it by stating that “methods requiring too much overhead for their relative benefits are ultimately not sustainable.” In a later column they describe the problem in more detail and conclude it with an observation that a small company, especially a startup, cannot ignore the up-front costs of SPI efforts even if the quality would prove free in the long run (Ward et al. 2001, p. 107). And as mentioned above, the goal of the study reported by Calvo-Manzano Villalón et al. (2002) was a reduction of effort and time spent on SPI implementation (p. 66).

A related goal for keeping the up-front costs low is the desire to gain a quick return on investments. This was another goal of the MESOPYME method developed by Calvo-Manzano Villalón et al. (2002); it is reported that even if the return on investments was substantial in other SPI efforts, the short term return on investments is more critical to SMEs because of their resource limitations. Richardson (2002) follows a similar line of thinking as she notes that small companies must be proposed improvements that take effect in as short a time period as possible for financial reasons. It is important to notice, however, that small companies are not alone with this problem and, for example, the study by Isaacson et al. (2001) summarized above reports that improvement programs are often perceived as laborious and having slow progress (p. 65).

As a last point on small companies and investments it is good to notice that the high cost reduction rates reported by some literature as a result of SPI programs in large companies are irrelevant for small companies. Namely, Ward et al. (2001, p. 105) note that small companies have had little chance to generate such overheads that cost reductions of 7 to 1 or alike would be possible. This means that small companies may be unable to justify investments in improvement by paybacks through cost reductions and, consequently, Ward et al. suspect that small companies are “far more interested in controlling the chaos inherent in growth” (2001, p. 105).

There is a long line of research on business process reengineering in large companies and many of these seem to report failures as natural parts of the efforts. For example, Caron et al. (1994) and Harkness et al. (1996) report failures, roadblocks, and difficulties all along the efforts that, however, could not stop the reengineering efforts. This approach, learning from experience, suggests that such companies have actually adopted the organizational learning and skill development approach suggested by Attewell (1992). As many do not acknowledge this need to learn from past failures, such failures are perceived as negative experiences that make organizations skeptical about process improvement efforts in general. Consequently, people become suspicious about available problem solving methods and whether or not they actually produce desired results instead of actually using them. This situation is well exemplified by, e.g., Nishiyama et al. (2000, p. 578).

The above-mentioned negative experiences have also been found important de-motivators for software practitioners concerning SPI efforts (Baddoo and Hall 2003, p. 31). Another de-

motivator for practitioners identified in this study was a lack of evidence of direct benefits of the efforts in practice (Baddoo and Hall 2003, p. 31) and it is not surprising that evidence of the benefits is among the most common motivators for SPI efforts (Baddoo and Hall 2002, p. 92). It has also been reported that the lack of IT adoption in small businesses is caused by the management unawareness of the clear anticipated benefits of the IT adoption in their companies and, on other hand, there is no social pressure to do so (Riemenschneider et al. 2003, p. 282). Thong (1999, p. 188) has also made a related observation about the situation in his literature summary: “there is a lower level of awareness of the benefits of IS and a lack of IS knowledge and technical skills in small businesses.”

The uncertainty about process improvement effort payback is succinctly illustrated by Mackay (1990). This study focused on software customization, and Mackay describes the perceived value of customization with an example curve of the net benefits gained from time spent on customizing a complex software package such as Emacs, a text editor (Figure 19). At the origin, no time is spent on customizing and no benefits are derived from customizing. An unsuccessful attempt takes time and may introduce problems resulting in a perceived (and actual) net cost. However, if the time spent on learning results in a single successful change, a net benefit is created. Additional customizations like init file changes or macros may result in additional benefits even if every customization tends to introduce also net costs when they are learnt. After a certain point, additional time spent customizing will no longer increase productivity and will begin to interfere with work. At the extreme, users who spend 100% of their time customizing are unlikely to obtain any net benefits and will most likely be faced with high net costs. Mackay emphasizes that the perceived costs and benefits are very much embedded in the individual user’s current context and work preferences and thus a change that is considered as a significant benefit to one person may be a waste of time for another. Mackay also notes that there is no easy way to evaluate costs and benefits because it is hard to estimate how long a customization will take, and on the other hand, the benefits are likely to be seen only after the customization is complete. Thus the user’s choices about when to customize are influenced by the *perceived* costs and benefits rather than the actual costs and benefits. Mackay summarizes the net benefit diagram as follows: “Each of the factors, costs, benefits and the threshold [when to customize], are variable. Some are under the users control, others are not” (Mackay 1990, p. 23).

The Mackay illustration of net costs and perceived benefits also fits SPI efforts well in general. An attempt to implement a process improvement will introduce a net cost in the form of lost time and possibly also investment costs, e.g. for tools or training. The potential benefits, on the other hand, tend to be things like improved performance or quality. Unfortunately, the benefits are usually hard to measure, which means that in many organizations the costs are easy to show but the benefits are not. A further problem is that in many cases the first unsuccessful attempts lead to abandoning of the effort before the benefits are seen since, as Mackay stated (1990, p. 22), “Costs and benefits cannot be evaluated easily, because users do not know in advance how long each type of customization [change] will take, nor can they fully appreciate the benefits until the customization [change] is complete.”

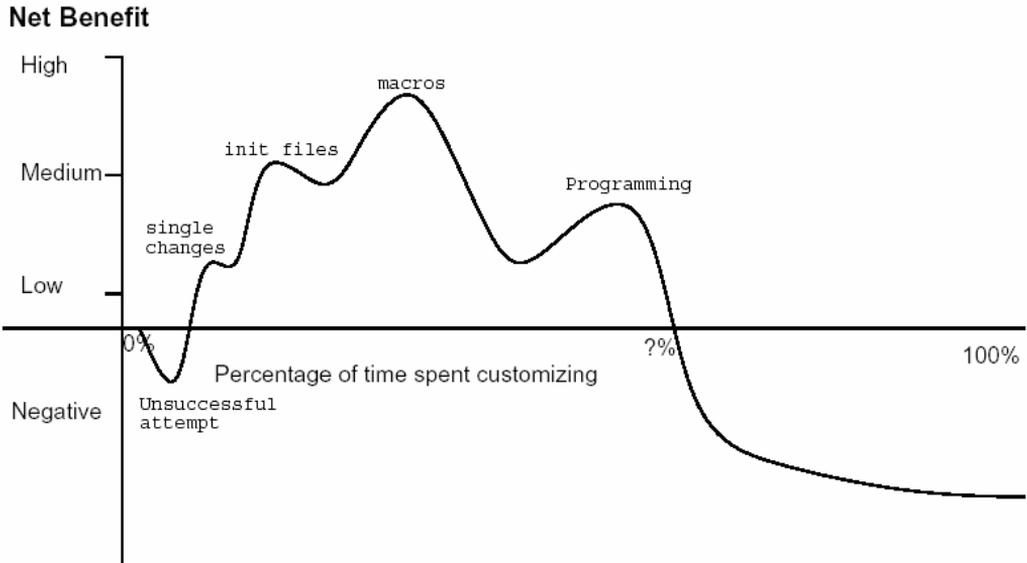


Figure 19. Net benefits vs. time spent customizing (Mackay 1990, p. 22)

To summarize the uncertainty about process improvement effort payback, it is caused by the fact that improvement costs are often tangible and easy to identify while the benefits are not. This problem is not made any easier by the observation that in many cases the benefits of improvements – and especially evidence of them – are unknown in companies. Thus, the basic options for alleviating the uncertainty are either to reduce the improvement costs or to make the benefits more tangible. As a final point it is good to bear in mind that the outcomes of an unsuccessful investment in a small company can be fatal: “Further, small businesses typically have less slack resources with which to absorb the shocks of an unsuccessful investment in IS adoption” (Thong 1999, p. 188). Thus SMEs have a concrete reason to avoid risky investments.

4.3. Application Domain for the Study

A fundamental requirement for the application domain selection in the present study was practical relevance to the local software industry. In the conducted state-of-the-practice investigation most of the companies developed some kinds of business applications even though many other kinds of applications were also produced. Thus focusing to business applications appeared to provide the best possibilities for evaluating the anticipated solution. Glass and Vessey (1995) studied existing application domain taxonomies and found a limited amount of foundational work on the topic. Thus it was decided to focus on an established application domain that would be fairly well recognized and the *administrative and business applications* domain from the Association for Computing Machinery (1998) classification was selected as the application domain for the present study. This domain provides a fairly broad coverage of different areas that can be expected to be useful for many companies in the anticipated adopter group.

The justification for focusing on such a seemingly familiar – and for some people even simple – application domain is found in business process reengineering articles and in the learning

perspective of technology adoption. For example, both Caron et al. (1994) and Harkness et al. (1996) emphasize the need to start process improvement activities from familiar projects, or as Caron et al. (1994, p. 247) put it: “CIGNA’s experience suggests the advantages of gaining reengineering experience and competence by starting with less complex initiatives.” Harkness et al. report very similar experience from the BOSE Corporation (Harkness et al. 1996, p. 366):

In the initial stages of the improvement program, the portfolio should consist almost entirely of improvement projects that are well understood and bounded. Such a portfolio provides a context for training and early success among improvement workers. However, as a base of knowledge is gathered, it is important to balance the portfolio of improvement work with processes that are large in scope, complex, not well understood, or which have yet to be designed. Avoiding these types of projects presents a high cost to the organization in terms of lost opportunities. However, addressing complex projects without a foundation of incremental improvement activity may present improvement teams with work they are not equipped to handle. The implication for IS managers is that momentum for higher-levels of improvement work comes from carefully matching improvement opportunities with existing levels of skill and expertise. Improvement work should be orchestrated such that current projects provide the necessary staging for more complex and higher impact process activity.

These experiences line well with the Attewell (1992) proposition that technology adoption is a learning activity. This, on the other hand, makes the well understood and bounded problems a fertile ground for learning the basics about process improvement and new technology adoption.

4.4. Summary

The conducted state-of-the-practice investigation provided evidence of the anticipated need for support in process improvement efforts in small organizations with low maturity. A sanity check for this organizational context for the study was done by studying literature, which revealed two characteristic problems in small organizations with low maturity – lack of resources (money, time, and expertise) and uncertainty about SPI effort payback. The application domain of the present study was then defined as administrative and business applications. This selection also reflects the basis for the whole process improvement effort of the present work: process improvement is not a project that has fixed starting and ending dates but a continuous process based on a desire to learn more and do better starting with well understood and bounded problems.

5. SOLUTION CONCEPT FORMULATION

This chapter focuses on formulating a solution concept to the problems identified in the previous chapter. According to Redwine and Riddle (1985, p. 191), this means converging on a compatible set of ideas that should provide a working solution to the identified problems. The following sections report the formation of the solution, solution characteristics, and a summary of the chapter. Overall, this chapter provides an answer to the first research question: *What are the desirable characteristics of a requirements engineering method that would make it easy for small organizations to adopt it?*

5.1. From the Problem to a Solution

The conducted state-of-the-practice study (Section 4.1) concluded with an observation that the companies needed support in RE in a holistic manner in all areas of RE. However, most of the topics suggested for RE research (Section 3.2.1) focus on limited topics, in many cases techniques to solve some individual problems. Broadening the search for viable solution alternatives brought up the personal views and perspectives on software process improvement of Watts S. Humphrey (2002) which provided two important insights from the point of view of the present study.

First, Humphrey (2002) notes the importance of assessments by reporting that as a result of a locally conducted assessment, the whole organization became aware of and understood the problems and why they had to be fixed. A closer consideration of the topic, however, brought up the fact that Humphrey has been dealing a lot with large and complex projects in which the importance of assessments is understandable. The focus of the present study, however, was on small and low maturity companies and thus the assessment approach appeared overkill. Instead, it was decided to give presentations, discuss the issues with potential adopters, and raise their awareness about the role of requirements related issues in software development without detailed assessment. It was reasoned that if the potential adopters had encountered problems in software development they would become interested in a potential solution and its adoption. This approach reminds that of Gause and Weinberg (1990) who report on a tunnel in Switzerland after which people often parked their cars and left the lights on leading to dead batteries and need for help. After describing alternative solutions, the actual solution – a sign in the end of the tunnel reading: *Are Your Lights On?* – is put forward with the following rationale (Gause and Weinberg 1990, p. 101):

If people really have their lights on, a little reminder may be more effective than your complicated solution.

The second interesting point made by Humphrey (2002, p. 66) was the following statement: “Method is critical and the disciplined practice of sound methods is the only way to learn to do consistently high quality work.” Since Humphrey has studied this issue extensively and shown the effect a method can have on the quality of the work (Humphrey 1998), construction of a method appeared a viable solution to the present problem. Considering Humphrey’s statement on methods in the light of the study by Introna and Whitley (1997) (p. 54), an idea was born to develop a ready-to-use method. Namely, a technically ready-to-use method was expected to address the problems identified in the problem analysis (lack of resources and uncertainty about SPI effort payback) since, by definition, a technically ready-to-use method does not need

adaptations in the adoption phase. Consequently, the need for resources was expected to be minimal and with a ready-to-use solution the uncertainty was also expected to be under control.

Humphrey's experiences in method development were further used to assess the proposed solution. Humphrey (1998) claims that a method is critical in learning to do high quality work, but he also reports that few Personal Software Process (PSP) trained engineers will continue using it without continued management interest and support (Humphrey 1998, p. 211). As an explanation for this Humphrey suggests the observation that the majority of students and engineers follow their pragmatic short-term interests and, based on his experience, the only generally accepted short term priority in software is coding and testing (Humphrey 1998, p. 211). In a more recent article, he summarizes the problems encountered with the PSP in one section (Humphrey 2002, p. 59):

It wasn't long before I realized that there are few managers or executives who naturally provide the kind of coaching, leadership, and support that the PSP requires.

In helping organizations introduce the PSP, we found that few engineers would use it in their work. It wasn't that they didn't believe the PSP would work or that it was particularly hard to use, but that the working environment discouraged them from using it. The PSP requires disciplined personal behavior and, as in many other fields, highly disciplined behavior requires a supportive working environment. Musicians have conductors, actors have directors, and athletes have coaches. In medicine and the sciences, there are rigorous laboratory courses and internship programs that demonstrate and instill the necessary skills and disciplines. There are also regulations, professional standards, and oversight bodies to ensure that the practices are consistently used.

In short, disciplined behavior is not normal or natural for most people. It requires extensive training, a supportive environment, and a change in management style.

Humphrey has also noted that students are not likely to use the PSP unless their professors use it also, or "practice what they preach" (Humphrey 1998, p. 217). This observation lines up well with one statement in the above quote: "It wasn't that they didn't believe the PSP would work or that it was particularly hard to use, but that the working environment discouraged them from using it." To put these statements in a larger context they both touch upon the *perceived ease of use* and the *subjective norm* of the method use. Green and Hevner (2000) report a more thorough analysis of the PSP and its diffusion to practice while the actual study behind this article is reported as a SEI special report (Green and Hevner 1999) going into considerable detail on the topic. The results of the study indicate that successful diffusion of innovative software development techniques – in this case the PSP – depends on four factors: developer involvement in the adoption process, timely and quality training, voluntariness, and champion push for the use of the technique (Green and Hevner 1999, p. 74). Thus, even without going into more detail about the PSP, it seems evident that technology transfer and diffusion of innovations research deserve a close consideration from point of view of the present study (cf. Section 3.2.5).

5.2. Characteristics of the Solution

In this section, the specific characteristics of the solution concept are introduced. The approach to discuss *characteristics* rather than *requirements* was adopted for two reasons. First, the solution should be characterized by the identified properties meaning that the degree of individual properties should not be maximized as such but an integrated solution should be developed where all the properties are present in a reasonably balanced set. Second, these characteristics are built around the diffusion of innovation characteristics so it seemed wiser to extend this set of characteristics rather than to develop a novel set of requirements.

The characteristics are grouped into three categories: the characteristics specifying the focus of the method, the diffusion of innovation characteristics, and other characteristics that were found important from the point of view of the present study. The descriptions of the individual characteristics are followed by a discussion about the relationships between the characteristics and how these characteristics are expected to contribute to solving the characteristics problems in small organizations with low maturity.

5.2.1. Specificity

Section 3.2.3 introduced the *domain engineering* research area suggesting that a domain specific – or strong – method is designed to fit and do an optimal job on one kind of problem. Based on these ideas, the rest of this section describes four different ways to focus on the anticipated solution.

3-Tier Architecture Focus. In lack of an application domain taxonomy that would provide definitive guidelines for categorizing applications to manageably bounded domains, a link between requirements and implementation is used. Namely, Pressman (2001b, p. 750) suggests that it is often appropriate to implement a client-server system from different subsystems including the user interaction/presentation subsystem (i.e., typically graphical user interface), the application subsystem (e.g., in a business application report generation based on numeric input, calculations, database information, and other considerations), and the database management subsystem (i.e., data manipulation and management required by an application). This approach provides a reasonable approach for the present work since this architecture has served as the basis for other more complex architectures (e.g., 3- and N-tier client server architectures) and, more generally, the user interface, application or functional unit, and database management subsystems are common elements in administrative and business applications which is the selected application domain of the present study (Section 4.3). However, for the sake of simplicity these three elements are collectively called here the “3-tier architecture focus.”

Small Application Focus. In general, the smaller the application is the smaller are the resources available for constructing development methods to produce it. Thus small application projects are expected to be interested in an easy to adopt method. In the present study, a small application refers to one that can be implemented with a project having one requirements engineer and less than half a dozen developers, a duration from six to twelve months, and a budget below 100 kEUR.

Small Team Focus. The small team focus is derived from the characteristics found in the state-of-the-practice study which suggested that in many cases the software development teams are small (Section 4.1). The justification for this characteristic is that potential method adopters in the present study are likely to be comparable with the previously studied companies. In practice, the small team refers to less than half a dozen people and one requirements engineer in every project; the requirements engineer is responsible for and in charge of the development of requirements documents as suggested in the *small application focus* characteristic.

Introduction Level Focus. The idea to focus on introduction level practices has little else but practical grounding. The companies are expected to lack basic systematic RE practices suggested widely in literature (cf. Section 4.1) and thus the goal is to get them in institutionalized use in industry.

5.2.2. Innovation Characteristics

The role of innovation characteristics in adoption was demonstrated by the PSP method in Section 5.1 and the concept in general was introduced in Section 3.2.5. In this section, the diffusion of innovation characteristics suggested by Rogers (1995) is summarized as it relates to the present study with two exceptions: *complexity* is replaced by *ease of use* (Davis 1989) and *observability* is split into two separate requirements, *result demonstrability* and *visibility* (Moore and Benbasat 1991). For ease of reference, these characteristics are also given adjectives that are later used to refer to them.

Relative Advantage. Rogers (1995, p. 15) defines relative advantage as “the degree to which an innovation is perceived as better than the idea it supercedes.” As explained in Section 3.2.5, Davis (1989, p. 320) suggests another name, *perceived usefulness*, for basically the same construct: “the extent they [people] believe it will help them perform their job better.” In the present work relative advantage is used to refer to, for example, the higher quality or faster implementation of a requirements document or faster implementation of the actual software in which the new RE method is expected to result. This characteristic will be referred to with *beneficial* adjective for the solution.

Compatibility. Rogers (1995, p. 15) defines compatibility as “the degree to which an innovation is perceived as being consistent with the existing values, past experiences, and needs of potential adopters.” The development work of the Moore and Benbasat (1991) measurement instrument identified problems with this definition due to the inclusion of *needs* in the definition. They analyzed the problem to be a confounding with relative advantage “as there can be no advantage to an innovation that doesn’t reflect an adopter’s needs” (1991, p. 199). Thus in this work compatibility refers to consistency between existing values and past experiences. The past experiences are often expected to be related to the use of various techniques, templates etc. in previous work assignments. This characteristic will be referred to with the adjective *compatible* for the solution.

Ease of Use. Davis (1989, p. 320) defines the *perceived ease of use* as “the degree to which a person believes that using a particular system would be free of effort.” This appears a more intuitive term than *complexity* used by Rogers (1995, p. 16): “the degree to which an innovation is perceived as being difficult to understand and use.” Another issue is the difference between *ease of use* and *ease of learning* which Davis (1989) resolved with a literature survey. Davis

summarizes this survey as follows (1989, p. 325): “Studies of how people learn new systems suggest that learning and using are not separate, disjoint activities, but instead that people are motivated by being performing actual work directly and try to *learn by doing* as opposed to going through user manuals or online tutorials.” Thus, from the ease of learning point of view it is important that a method is detailed enough that it is possible to use it by following the provided instructions. Humphrey (1998, p. 216) puts this clearly as a lesson from the PSP and TSP: “When given a clearly structured set of tasks and a defined process for handling them, engineers will generally follow the process. This assumes that they know how to follow a defined process.” Humphrey continues by stating that for students to consistently use sound methods, much more detailed processes must be provided to them than has been done in the past. This characteristic will be referred to with the adjective *easy to use* for the solution.

The ease of use is especially important in small organizations where, e.g., a complete requirements development activity may be conducted only once a year or even more seldom. Thus there is a high probability of forgetting the method and its use during the inactive time resulting in a need to relearn or invent the method again every time it is used.

Trialability. Rogers defines trialability as “the degree to which an innovation may be experimented with on a limited basis” (1995, p. 16). Later on he claims that “new ideas that can be tried out on the installment plan are generally adopted more rapidly than innovations that are not divisible” (1995, p. 243). A limited experimentation can be implemented either by testing individual parts of the method alone or by testing the whole method in a limited number of projects. In both the cases trialability allows the actual decision about a wider scale testing or adoption to be made on company specific experiences of the method. Notice that in the present study trialability and trial use are used interchangeably; this characteristic will be referred to with the adjective *triable* for the solution.

Result Demonstrability. Result demonstrability is included in the study by Rogers (1995) within the *observability* characteristic. However, the Moore and Benbasat (1991) study experienced problems with the construct validity of observability and as their analysis revealed the complexity of the structure, it was split into *result demonstrability* and *visibility*. Consequently, they defined the first part as follows: “the **tangibility** of the results of using the innovation, including their **Observability** and **Communicability**, and was labeled **Result Demonstrability**” (Moore and Benbasat 1991, p. 203). This characteristic will be referred to with the adjective *result demonstrable* for the solution.

Visibility. The analysis by Moore and Benbasat (1991, p. 203) of the Rogers (1995) observability construct reveals the idea that “the more a potential adopter can see an innovation, the more likely he is to adopt it.” Thus, in addition to the fact that results of the method use should be observable, also the innovation itself should be visible. This characteristic will be referred to with the adjective *visible* for the solution.

5.2.3. Other Characteristics

The third group of characteristics for the solution emerged from the conducted state-of-the-practice investigation (Section 4.1), the literature survey (Section 3) and early industry contacts. The identified three characteristics – comprehensiveness, readiness-to-use, and adaptability – have not been studied much as such even though some indication of the importance of these

characteristics can be found in literature. Since these characteristics will also be used to characterize the future solution, an adjective is given for each characteristic in the following sections.

Comprehensiveness. The comprehensiveness characteristic means that the anticipated method should address all the key areas in RE. However, it is clear that the scope of the solution is intimately connected with the problem at hand and thus little concrete advice on concretizing this characteristic can be found in literature. However, there are some articles on this issue that build on anecdotal evidence. For example, Mead (2000) addresses this topic by focusing on the need of both requirements *engineering* (or development, cf. Figure 3, p. 28) and requirements *management*. In this article, under the subtitle “You can’t have one without the other”, Mead reports observations that many organizations have only addressed either the development or management part of RE and, consequently, serious deficiencies in the overall RE have been identified. These observations appear consistent with the Nishiyama et al. (2000, p. 578) experience that people may know some parts of the problems but the entire problems are not understood. A more general viewpoint to this issue can be found from the differences between novice and expert approaches to problem solving. Namely, Mackay and Elam (1992) report that novices approach problems with a bottom-up procedure without a comprehensive plan and relying on the surface characteristics of the problem while experts in general are inclined to approach them from the top down “and are able to group problems with the same underlying structure” (Mackay and Elam 1992, p. 151). Thus it can be concluded that having a comprehensive plan could ease the introduction of RE practices in companies without expertise in RE.

Even though a comprehensive approach is suggested, it must be noted that maximizing this aspect is unlikely to work well in practice. Namely, increasing comprehensiveness is likely to increase the method complexity at the same time and, consequently, tax the ease of use of the method. There seems to be a kind of consensus in the RE literature about the RE activities in general since, for example, Kotonya and Sommerville (1998, p. 35), Thayer and Dorfman (1997, p. 1-2), and Wieggers (1999, p. 19) suggest fairly unanimously – and compatibly with the Mead (2000) approach above – the following activities for RE: elicitation, analysis, documentation, validation, and management (Figure 3, p. 28). However, the conducted literature survey on RE approaches (Table 24, p. 123) clearly demonstrated that methods with such a scope are not yet common and thus an approach addressing all these key activities in RE can be claimed a comprehensive one. This characteristic will be referred to with the adjective *comprehensive* for the solution.

Readiness-to-Use. The readiness-to-use characteristic tries to eliminate the need for adaptations before a method can be adopted. This characteristic was developed based on the Introna and Whitley (1997) article discussing limitations of methods (p. 54). Namely, Introna and Whitley (1997, p. 236) claim that in order for tools to be used in practice they need to be available or *ready-to-hand*:

Second, we will attempt to explain the way humans interact with the world by means of ready-to-hand tools. This discussion will show that tools are used only if available (ready-to-hand) in the world of doing. If a methodology is not ready-to-hand it will break down and be ignored in the pragmatics of getting the job done.

Since availability as such is missing from the earlier identified set of solution characteristics, it was deemed as a potentially important new characteristic for the anticipated solution. However, a closer study revealed that this term was suggested in Heidegger (1927) (p. 24) and different interpretations of the book and the term have been suggested (e.g. Winograd and Flores 1986; Dreyfus 1991). Further, the term does not appear to belong to everyday English language since many native speakers have questioned its meaning and it is not defined in all English language dictionaries. Consequently, the term ready-to-hand was further developed into *ready-to-use* since in the present study availability could not be considered a real issue for the method adopters when it refers to the ability to locate and get access to the anticipated method. However, a related problem with availability is to get the anticipated solution adapted to the target environment to the degree that it can be taken into production use, i.e. that the solution is ready to use in a company. Thus the overall goal of the ready-to-use characteristic is to assure that the method is by design ready to be used in practice (cf. p. 34). Provided that the intended use matches the method focus, a method adopter should not need to adapt, refine, or supplement it in any way before the method is adopted.

The readiness-to-use of methods has not been discussed extensively in literature before but a closer look is provided into two books and one article related to this topic. The closest match with the present study is provided by Leffingwell and Widrig (1999) who report a common request they had often heard in their classrooms: “Just give us a single generic process what we can start with. We know it’s not that simple, but we’ll be happy to modify it as necessary to *our* project, but we need a more prescriptive starting point, a step-by-step process so that we can better apply what we have learned. *Just tell me how to get started!*” (emphasis original) (Leffingwell and Widrig 1999, p. 389). Consequently, the book closes with an initial process for beginners as a recipe to get started with RE (Leffingwell and Widrig 1999, p. 396-399). Another example of a process developed to a ready-to-use point is provided by Watkins (2001) which describes an off-the-shelf software testing process in three parts. The first part covers the traditional view of the components comprising a software testing process; in the second part a series of case studies are reported showing how the components introduced in the first part have been implemented in practice; and the third part includes a set of standard testing document templates, proformas, and checklists. These artifacts are also available from a website “so that they can be used immediately without modification or customized to reflect the particular requirements of any organization” (Watkins 2001, p. 3). As the final example, Humphrey (1998) touches upon this issue with the observation that much more detailed processes must be provided for students to help them use methods consistently. This observation was reported to support the ease of use characteristic (p. 91) but it is also related to readiness-of-use since the adoption phase may prove quite different from the continued use of a method. This characteristic will be referred to with the adjective *ready-to-use* for the solution.

Adaptability. The adaptability characteristic is needed to supplement the readiness-to-use. The need for adaptability has been raised in a number of articles and there are some experience reports on adaptations in practice, but an overall assessment of the situation is that, in general, the need for adaptability is mostly based on anecdotal evidence. Nidumolu and Knotts (1998) studied the effects of customizability on perceived processes and competitive performance in software firms and conclude as a key finding from their study that their hypothesis “Increases in process flexibility are positively associated with increases in perceived competitive performance” was supported both by development and marketing manager groups (Nidumolu and Knotts 1998, p. 126). This study is reported to have several limitations starting with measurement problems with the performance. A technical viewpoint to adaptability is provided

by Ward et al. (2001) who claim in their column that small software companies have a constant issue in changing the existing processes to match changing circumstances. This is very closely related to the scalability issue in the traditional sense of growing with the company but addresses also the fact that in a small company the changes in application domain etc. can be very abrupt. As a last paper emphasizing the need for adaptability the Richardson (2002, p. 112) paper claims that *flexibility and easy to use* are central characteristics for an SPI model targeted for small software companies. Unfortunately, the only justification for this property is the following: “The SPM [proposed solution] was developed based on nine specific characteristics reflecting financial constraints and the availability of fewer people (National Competitiveness Council 1999)” (Richardson 2002, p. 108).

The experiences of customization behavior in practice are mixed. Lycett et al. (2003) report their experiences of migrating agile methods to standardized development practice in a large company. In their case, the overall framework was not developed to be a prescriptive one but “an aid in tailoring the process to context and in educating the people involved. Consequently, developers actively revisit the case in each development iteration” (Lycett et al. 2003, p. 83). Mackay (1990; 1991), on the other hand, studied customization among IT users and claims that “since time spent customizing is time spent not working, many people do not take advantage of the customization at all” (1991, p. 153). Mackay further reports that users need a reason for customization, which usually means external events forcing them to stop and reflect on their behavior. She elaborates this issue further and reports that the reason for customization is often a desire to maintain a stable interaction with the software rather than to take advantage of new features. This conclusion is consistent with the Chau and Tam (1997) finding that as long as organizations are uncertain about their ability to adopt new technology, they would rather maintain their current systems.

In short, the prior research findings on adaptability are rather initial than conclusive ones. In the workshop conducted to introduce an early draft of the method developed as part of the present work, the actual reasons for expecting adaptability were not addressed more closely, either. However, it is easy to believe that companies with world class products (i.e., with their own market niche) are tempted to believe that their development needs are also unique and that they, consequently, would need to adapt almost any standard method to their specific needs. Companies focusing on tailored software, on the other hand, are used to developing software to the specific needs of the customer at hand and, therefore, may find it questionable from the business point of view to rely on methods that do not lend themselves to similar situation specific adaptability. Considering the major changes, for example, in operating systems (e.g., VAX/VMS, Unix, Linux; DOS, OS/2, Windows) and application platforms (e.g., mainframes, client/server, personal computers, Internet) that have taken place in the last two decades, it seems only natural that few experienced people are willing to adopt a strictly constraint method as the one anticipated in the present work (Section 5.2.1) unless it was also adaptable. Finally, since the anticipated method is targeted for introduction level in RE, it is only natural that as experience is gained in RE, adaptations to the already adopted practices are made. Therefore, to ease to introduction of the anticipated method, it is necessary to allow for adaptations and support them rather than constrain them. This characteristic will be referred to with the adjective *adaptable* for the solution.

5.2.4. Discussion

In this section, thirteen characteristics were developed for the anticipated RE method based on the problems identified in the previous chapter. Table 17 presents these problems as columns and the characteristics as rows; in each cell there is a character indicating the degree to which the given characteristic should address the shown problem. A direct contribution to a problem solution is identified with the character ‘●’, an indirect contribution with the character ‘○’, and lack of concrete contribution is identified with a dash; the previous sections provide justification for the suggested contribution estimates.

Table 17. Identified problems and the degree to which specified characteristics are expected to contribute to solving them

Characteristic \ Problem	Lack of money	Lack of time	Lack of expertise	Payback uncertainty
3-Tier Architecture Focus	-	●	●	○
Small Application Focus	-	●	●	○
Small Team Focus	-	●	●	○
Introduction Level Focus	-	●	●	○
Relative Advantage	-	-	●	○
Compatibility	●	●	●	○
Ease of use	○	●	●	●
Trialability	-	-	○	●
Result Demonstrability	-	-	-	●
Visibility	-	-	-	●
Comprehensiveness	●	●	●	○
Readiness-to-Use	●	●	●	●
Adaptability	-	-	-	●

Table 17 deserves two further notes. First, the problems are not quite orthogonal ones and especially lack of time or expertise can be compensated to some degree with money and vice versa. In this case, no such trade offs are anticipated but the primary relationships are considered. Second, the characteristics are not orthogonal ones but represent to some extent contradicting characteristics and, on the other hand, characteristics that contribute towards the same goals. For example, comprehensiveness and ease of use are clearly contradicting requirements and even a great deal of detail that should make a method easier to learn can actually decrease the ease of use (this is demonstrated, e.g., by PSP, Section 5.1 on p. 89). Trialability, on the other hand, has a comparable goal to readiness-to-use but it does not account for the costs of actual adoption and possibly needed adaptations. That is, a trial use is generally done to demonstrate the benefits of the concept but actual use may require changes that are not considered at this time; readiness-to-use focuses on this aspect and tries to assure that there are no hidden surprises in the actual adoption phase. There are also characteristics that are expected to contribute to another characteristic. For example, readiness-to-use, adaptability, and compatibility are expected to contribute to the ease of use of a method. Overall, each characteristic addresses a unique aspect justifying its inclusion. Thus the developed set of

characteristics presents a practical set of characteristics and an implementation should provide a reasonable balance between the whole set of characteristics.

5.3. Summary

This chapter focused on the solution concept formulation. First it was decided to develop a method to address the identified problem and then characteristics of the desired solution were developed. These characteristics, and respective requirements, were developed from domain specificity, diffusion of innovations literature, and combining findings from the conducted state-of-the-practice investigation, literature survey, and early industry contacts. A complete list of the characteristics includes the following items:

1. 3-Tier Architecture Focus
2. Small Application Focus
3. Small Team Focus
4. Introduction Level Focus
5. Relative Advantage
6. Compatibility
7. Ease of use
8. Trialability
9. Result Demonstrability
10. Visibility
11. Comprehensiveness
12. Readiness-to-Use
13. Adaptability.

It was also noted that these characteristics are not quite orthogonal but present a set of characteristics for which a working balance needs to be found.

6. THE BARE METHOD

This chapter describes the BaRE method, which was developed based on the problems and characteristics identified in Chapters 4 and 5. Section 6.1 approaches the method from a constructive research point of view and presents the key components with the rationale for their selection. Section 6.2 presents an overview of the method itself and Section 6.3 summarizes the limitations of the method. Section 6.4 describes alternatives and extensions to it before closing the chapter with a summary in Section 6.5. The history of the method construction is outlined in Section 1.5 and the method is documented in full in Nikula (2002b). Overall, this chapter provides an answer to the second research question: *Is it possible to develop an easy to adopt method for requirements engineering?*

6.1. Method Components and Relationships

In this section, the key components of the BaRE method are described. The method is introduced based on the Avison and Fitzgerald (2003) approach suggesting that a method should possess five attributes: phases, techniques, tools, training, and philosophy (Section 3.3.1, p. 68). From the point of view of the adopter, the key philosophical principles of the BaRE method are the following:

- In a small organization and small project environment, lightweight approaches can provide valuable and sufficient structure to working practices.
- Evaluating the benefits of a packaged solution is easier than evaluating the benefits of individual practices.
- Documented requirements and change requests are the basis of systematic requirements development and management.
- Systematic processes for requirements development and management are the basis for objective and efficient handling of all requirements and change requests.
- An automated tool can help to manage and improve the quality of requirements and change requests.
- Some basic requirements engineering concepts and techniques should be understood before actual requirements engineering work is undertaken.

These central principles are reflected in the solution development in the following ways. First, a packaged solution called method was decided to be developed based on the ideas of lightweight or agile software development approaches. Second, from the RE point of view four central elements were identified: documents, processes, requirements development and management areas. Third, automated tool support and need for training appeared viable elements of the solution. Since the RE community has not established a unanimous understanding between the terms *requirements engineering* and *requirements management*, it was decided to keep these areas explicitly separated resulting in altogether five components in the method. The first of the components is a requirements document template which reflects the importance of documented requirements but addresses also the *techniques* attribute suggested by Avison and Fitzgerald (2003). The second and third components are the requirements development practices and requirements management practices respectively (covering both *phases* and *techniques* of the Avison and Fitzgerald approach). Separating the practices between the two areas in RE presents

some overlap in comparison with the reference models but, as stated above, it appeared justified to avoid misconceptions caused by the terminological issues in RE. Finally, tools and training attributes are included as such as the fourth and fifth components. These components are summarized in the following list and the rest of this section introduces each of them in more detail; the section is closed with thoughts on how to introduce these components in a company.

1. Requirements document (RD) template.
2. Requirements development practices.
3. Requirements management practices.
4. Tool support for requirements management (RM).
5. Training.

Requirements Document Template. The first component is the requirements document template which refers to project-independent implementation of RD. The importance of documenting requirements is brought up by, for example Gause and Weinberg (1989, p. xvi) stating that “The document is nothing; the documenting is everything.”⁴ In the present work it is assumed, however, that also a produced document has some value since Suchman (1989, p. 4) claims that practice is mediated by artifacts. A third viewpoint on the role of requirements documenting is provided by Sommerville and Sawyer (1997, p. 44) stating that this is one of first things an organization should do when improving RE practices. This perception appears to be shared by many others since requirements document templates are among the oldest and best documented parts of RE – for example, Dorfman and Thayer (1990) present 20 standards on RDs and the oldest of these dates back to 1976 (Federal Information Processing Standards 1976). Most books on RE suggest that an RD should be developed even though the degree of formality and details vary (cf. Sommerville and Sawyer 1997; Robertson and Robertson 1999; Wiegers 1999; Maciaszek 2001; Young 2001). The state-of-the-practice surveys indicate that RD-related practices are not very impressive and belong to the ones needing improvement (Kamsties et al. 1998; Nikula et al. 2000a), which supports a more general claim that more than 85% of all enterprises have inadequate software documentation methods (Jones 1996, p. 228). The importance of the RD derives from the many roles it has: it can be used as a basis for software development contracting (Robertson and Robertson 1999, p. 137), for external reviews (Boehm 2002, p. 66), for reuse in other projects (Czarnecki 2002, p. 433), as well as a standard medium to communicate about requirements between customers, users, managers, and developers (Sommerville and Sawyer 1997, p. 38). Due to these multiple roles an RD template can, in practice, consist of multiple related templates. In the present work, the RD template is developed from the software development point of view and targeted especially for the developers to be used as a memory aid. This approach follows from the observation that programmers are influenced by what they are asked to do (Weinberg and Schulman 1974) which is summarized by Gause and Weinberg (1989, p. 1) as follows:

- If you tell what you want, you’re quite likely to get it.

⁴ This statement is adapted from the Dwight Eisenhower’s statement: “The plan is nothing; the planning is everything.”

- If you don't tell what you want, you're quite unlikely to get it.

In the present study, the goal is to increase the likelihood of getting what is requested by providing assistance in memorizing the requests with documented requirements.

Requirements Development Practices. Requirements development practices are the second component. They consist of requirements elicitation, analysis, documentation, and validation tasks (Sommerville and Sawyer 1997; Thayer and Dorfman 1997; Kotonya and Sommerville 1998; Wiegiers 1999). In the following requirements, development will be considered as covering these tasks but they will be approached from the viewpoint of processes and techniques as suggested by Kotonya and Sommerville (1998) and supplemented with checklists. Considering the design feature categorization shown in Figure 20, the present study focuses on the *Essential* and *Expected* features of a software rather than the *Gee Whiz* ones. This approach is based on the idea that these two feature categories are more important in running a steady and predictable business while the third category may operate on different principles and risk levels leading to a situation where basic and systematic practices may not be the best ones to follow.

The definitions of the three feature categories are as follows (communication with Donald C. Gause):

Essential features define the very essence of the product. As an example, a travel alarm clock must be portable, keep reasonably accurate time, and provide an alarm feature, among other things. If any one of these features is missing the public would not accept the product as a travel alarm clock. Essential features do not add value to the product.

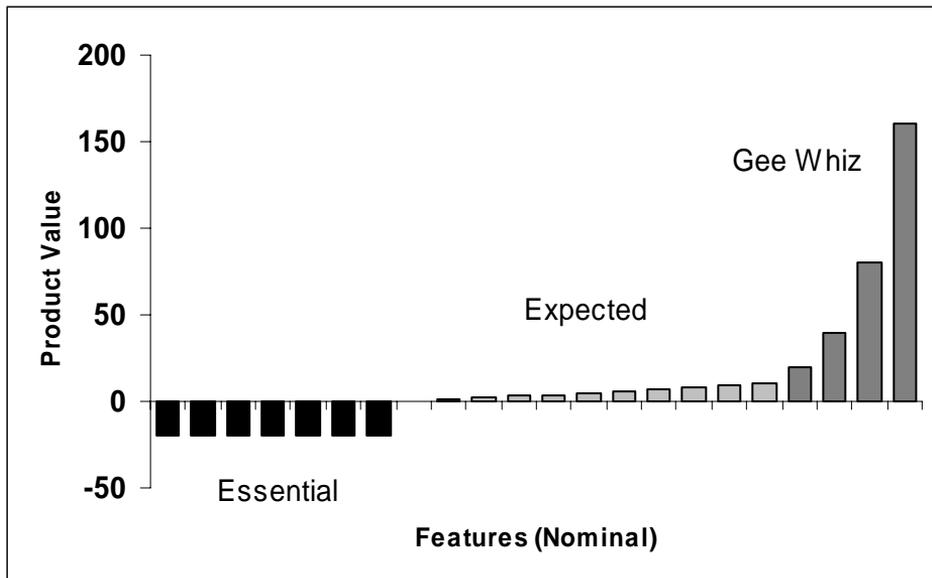


Figure 20. Three categories of design features (Gause 2000)

Expected features are parts of the general class of products but not to the point that they form the very essence of the product. Battery powered or wind up, digital or analog, acoustical alarm or flashing light could be thought of as expected features. Each of these features adds positive marginal value to the product.

Gee whiz features are the genuinely surprising and useful features. When we see them, we say to ourselves, "Gee whiz, why didn't I think of that?" These are the features that enable the public to differentiate the new product from the others in the market place and, as such, add substantial value to the product. A GPS (global positioning system) travel alarm, which resets itself to the local time while simultaneously displaying the clock-positional local time and time-at-home, would be a product with gee whiz features.

Requirements Management Practices. The third component is the requirements management practices. RM begins with a baseline RD and keeps the changes under control during the rest of the software development (Sommerville and Sawyer 1997; Kotonya and Sommerville 1998; Wiegers 1999; Lauesen 2002). As with requirements development, RM will be approached from the viewpoint of processes and techniques supplemented with checklists. There are two reasons for having RM as one of the key components. Firstly, RE can be divided hierarchically to requirements development and management activities (Wiegers 1999, p. 19), and the success of either activity without the other can be questioned (Mead 2000). Secondly, the need for RM is evident since it has been claimed that "It is often the case that more than 50% of a system's requirements will be modified before it is put into service" (Kotonya and Sommerville 1998, p. 113). It should also be kept in mind that the difference between requirements and change management may be marginal or even non-existent in some companies (p. 29).

Tool Support for RM. The fourth component for the method is tool support for RM. The tool support should preferably cover the whole RE, but since the requirements development phase still lacks proven solutions, it was not justified to suggest it in the method. For example, Damian et al. (2000, p. 28) report that face-to-face group meetings for requirements negotiation do not perform better than those using video conferencing and computer support while Kotonya and Sommerville (1998, p. 41) report that "In practice, however, we have found that putting a computer between the requirements engineering and the system stakeholders often inhibits the elicitation process." Pohl (1994, p. 252) also suggests the use of, e.g., decision support systems and computer-supported cooperative work systems to support RE, but these ideas seem quite distant from the basic RE point of view. However, the use of tools to support RM appears to be unanimously encouraged (Sommerville and Sawyer 1997; Kotonya and Sommerville 1998; Leffingwell and Widrig 1999; Robertson and Robertson 1999; Wiegers 1999; Hooks and Farry 2000; Maciaszek 2001; Young 2001; Lauesen 2002). It has also been reported in a recent study (Matulevicius 2004) that RE tools provide better support for the RE process than general office and modeling tools resulting in a higher quality requirements document. Thus tool support for RM appears a justified component in an RE method.

Training. The fifth and last component is training. The real issue behind training is the need for skilled personnel (cf. The Standish Group 1994; Keil et al. 1998; Constantine 2001; Hofmann and Lehner 2001) and the ways to support the development of the knowledge and understanding required from real experts. Introna and Whitley (1997) as well as The Standish Group (1994) consider understanding a prerequisite for the successful use of a method; Introna and Whitley even suggest extended formal apprenticeships as a way to gain such understanding. Another

perspective to this issue is the one provided by Cockburn (2002) and Beck (1999b) with their three levels of method users which both can be considered also as levels of understanding (p. 50). One benefit of having these levels defined is that they make it possible to develop a strategy to increase the level of understanding. Considering the ease of use characteristic it is clear that documentation that allows for independent study, possibly supported by training and consultation of a topic area expert, provides a viable solution candidate.

Relationships. Despite the acknowledged preference of incremental change, it is suggested that these five key components may need to be introduced simultaneously as a radical change in a company due to the interconnections they have with each other. Namely, addressing any single component may appear useless in the absence of some other component. For example, adopting only an RD template leaves people wondering how to actually complete it – a problem that is addressed by the requirements development practices. Adopting requirements development practices, on the other hand, may prove of little use unless their limits are known with a decent general understanding of RE, which can be developed with training. Even an RD template and requirements development practices with appropriate training may fall short in practice unless requirements management is also addressed. Finally, in any project of a little more size there is a risk that so many requirements and change requests are proposed that managing them manually becomes an impossible task, and tool support for RM is needed. This set of relationships between the components is not an exhaustive one but demonstrates the need to address all the components in the method. This reasoning follows the same line of thinking as Beck (1999b, p. 149):

The full value of XP will not come until all the practices are in place. Many of the practices can be adopted piecemeal, but their effects will be multiplied when they are in place together.

Beck continues to admit that this is actually just a hypothesis and no evidence for the claim is provided. Beck closes discussion on this topic by noting that significant gains can be achieved even from parts of XP even though he believes that much more can be achieved by adopting all the pieces.

6.2. Method Overview

This section provides an introduction to the BaRE method itself. First a look at the key elements in practice is provided (Section 6.2.1), then the requirements document template (Section 6.2.2) and the requirements development process (Section 6.2.3) are considered more closely before the section is closed with descriptions of the method documentation (Section 6.2.4) and its use (Section 6.2.5). The method documentation section includes the table of contents of the BaRE Guide (Figure 24, p. 110).

6.2.1. Key Elements in Practice

The BaRE method is built around the RD template and its use is explained in the BaRE Guide. As a practically oriented method, the idea is that a method user takes the RD template and starts developing RD with the instructions provided in the BaRE Guide. The RD template includes placeholders for all the normal contents of an RD for an administrative or business application while the BaRE Guide describes the practices (processes, techniques, and checklists) for the

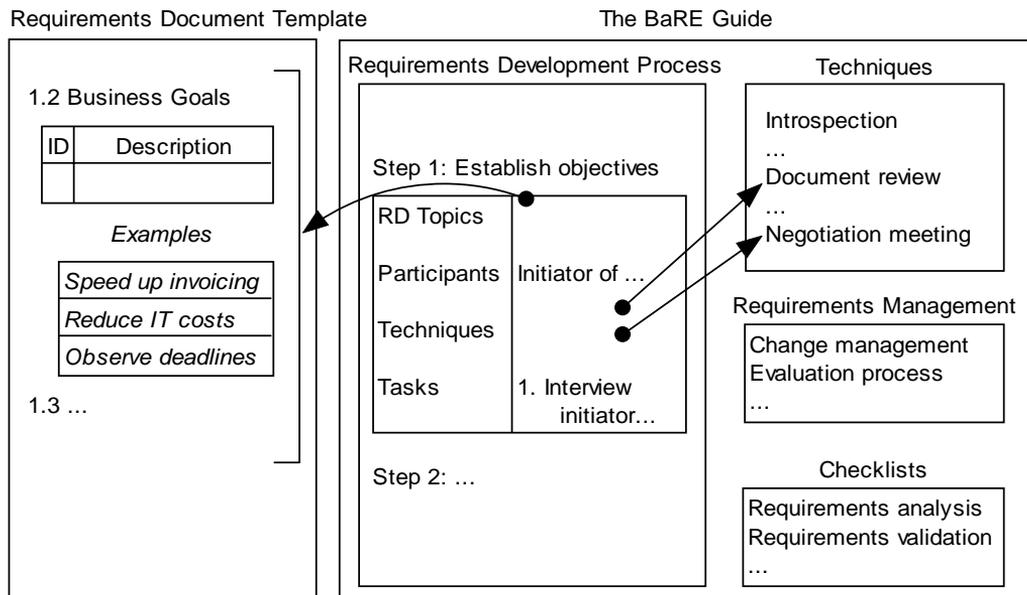


Figure 21. The BaRE method

method. These elements are illustrated in Figure 21. The actual RD template and requirements development process are shown in Appendices 4 and 5 of this thesis.

As Figure 21 shows, the general structure of the RD is that each topic has its own table with ready-made column titles. For example, in Figure 21 the business goals are documented in Section 1.2 of the RD and the table for them has two fields – a description and a unique identifier. This table approach is comparable with the requirements list, which is known to have shortcomings (Kulak and Guiney 2000, p. 15). The tables were chosen since they provide a simple, consistent, and compact way of presenting requirements with concise language – ideas advocated by, e.g., Sommerville and Sawyer (1997, p. 147) and Kovitz (1998, p. 149). It is, of course, clear that the tables do not automatically result in these properties but they do have an intuitive appeal for the properties. Each requirement table is supplemented with examples that illustrate common requirements for each category and provide examples of an appropriate level of detail, as well as the writing style. However, since the need for examples varies a great deal they are written as hidden text so that the user can choose whether he/she wants to see them or not.

The BaRE Guide includes detailed practices how to actually carry out requirements development and management including process descriptions, short technique overviews, and initial checklists. The development process includes nine steps, and each step suggests the RD topics to address in it. The steps and RD contents are synchronized so that the first step addresses only topics in the introduction chapter of the RD. To make the completion of the RD even easier, the participants, suitable techniques, and tasks to complete in each step are suggested. Thus the method includes guidance through the whole requirements development process even though the user is free to follow the process at his/her own discretion. The

requirements development process will be described in more detail later on in this section. The other processes in the guide include the RM process, which in practice is limited to an evaluation process for requirements and change requests. Due to the desire for ready-to-use processes, the requirements development process is an extensively modified version of the process suggested in Kotonya and Sommerville (1998, p. 35), while the evaluation process, on the other hand, appears fairly ready so it is only a slightly adapted version of the process described in Wiegers (1999, p. 240). Both processes are described in such detail that anyone who understands the software process basics should be able to apply them in practice.

The BaRE techniques are based on standard RE techniques explained in detail in many sources (e.g. Sommerville and Sawyer 1997; Kotonya and Sommerville 1998; Leffingwell and Widrig 1999; Robertson and Robertson 1999; Wiegers 1999; Lauesen 2002). From these the most suitable ones were selected on the basis of their fit for the purpose and ease of use. Keeping the number of techniques fairly small was important to keep the method simple. The BaRE Guide includes a short description of each technique and a reference to a more detailed explanation. Since most techniques are simple and intuitive, a short description is expected to suffice in most cases. For example, “electronic requirements” is hardly a self-explanatory name for a requirements elicitation technique, but a few lines of description relating it to using the Internet as an information source for requirements, and email as a way to communicate and elicit requirements from stakeholders is surely enough for most software engineers to start using this technique. However, if a reader wants a more detailed account on the technique, he/she is referred to a more complete description of the technique, e.g. Robertson and Robertson (1999, p. 99) in the case of the electronic requirements.

The basic techniques for RM include unique identification, versioning, baselining, and change management. With these techniques each item can be separately identified and the versions can be differentiated. The above-mentioned evaluation process is suggested to be used before baselining, and a single change request process is proposed to control changes. If change management procedures have already been established for a project, they should be retained instead of the BaRE mechanism. However, basic document templates are provided for configuration management as well to avoid a need to invent them just for managing requirements changes.

As the last item in Figure 21, a note on checklists is provided. Checklists are often advocated in literature due to their effectiveness and simplicity (Sommerville and Sawyer 1997; Wiegers 1999). Consequently, the BaRE Guide includes initial checklists for requirements analysis, validation, and different reviews.

The need for tool support in RM was evident both in the literature survey and in discussions with industry representatives. However, due to the resource constraints in this project no attempt was made to implement software support for RM, and therefore, other solutions were sought. Discussions with the industry quickly revealed that the needs of the companies varied to a great extent – some appeared interested in plain document templates while some were interested in commercial tools or automated freeware tools. Consequently, it was decided to use these three options in the evaluation projects and try to clarify both the rationale for each tool selection and the suitability of the selected tools for practical work. The document templates have been implemented with MS-Word and the descriptions of the templates and their implementation below refer to these templates – i.e. the BaRE document templates. It is worth mentioning that in addition to standard text processing capabilities, the hidden text feature is the

only special feature used in the implementation. The second option of using some automated tool includes, in principle, the use of freeware RM tools (e.g. C.A.R.E (SOPHIST Group 2002), REM (Toro 2002), or *sfrm* (sfrm 2002)) and possible company specific tools developed for this purpose. The last option of commercial tools refers to standard RM tools as listed, e.g., in INCOSE (2003).

The training issue for the method was solved by developing a short guide for the method and supplementing it with a more extensive guide including more background information on RE. Thus even a beginner is provided with an easy access to basic information that should help in getting started with the use of the method and the RE improvement effort.

6.2.2. Requirements Document Template

The standard approach to RD development today is to use an existing document template as the starting point (cf. Sommerville and Sawyer 1997; Kotonya and Sommerville 1998; Leffingwell and Widrig 1999; Robertson and Robertson 1999; Wiegers 1999; Maciaszek 2001). This approach has been criticized by claiming that a prefabricated structure and contents are harmful to the quality of an RD (Kovitz 1998, p. 129, 284) and, on the other hand, it has been claimed that a user’s manual makes an excellent software requirements specification (Berry et al. 2004). However, in the present work the mainstream approach is adopted and an RD template is provided since this appeared a familiar and straightforward approach. The BaRE RD template covers three main areas – introduction, overall description, and specific requirements – that are shown in Figure 22 with the second level headings supplementing them. The actual RD template includes also third level headings for the cases where more detailed descriptions are needed.

1 Introduction	2.4 Users
1.1 Customer Problem	2.5 Problem Domain Description
1.2 Business Goals	2.6 System Requirements Allocations
1.3 Stakeholders	2.7 Constraints
1.4 Product Purpose	2.8 Assumptions
1.5 Product Position Statement	2.9 Dependencies
1.6 Document Overview	3 Specific Requirements
1.7 References	3.1 Performance Requirements
2 Overall Description	3.2 Data Requirements
2.1 Product Context	3.3 Software System Attributes
2.2 Product Perspective	3.4 Other Nonfunctional Requirements
2.3 Product Functions as Use Cases	3.5 Design Constraints

Figure 22. The requirements document contents

Common tasks in administrative and business applications include supporting or enforcing a workflow, or informing about the state of the business. These kinds of applications are in general based on user interactions and include a data store, user interfaces, and interfaces to other systems. All these topics are standard RD topics, so the ways to document them have been adopted from other RD templates: user interfaces (Kovitz 1998), dialog maps (Wiegers 1999), user data (Wiegers 1999), workflow (Cheesman and Daniels 2000), and other system interfaces

(ANSI/IEEE Standard 830 1998). In addition to these key topics, the RD contents in general have been selected from eight different RD templates with the domain properties in mind (STRÍ TS2 1991; IEEE/EIA 12207.1-1997 1997; ANSI/IEEE Standard 830 1998; Kovitz 1998; Leffingwell and Widrig 1999; Wiegers 1999; Pressman 2001a; Robertson and Robertson 2001). The reason for considering so many templates are their different foci – e.g. the IEEE standard 830 is the de facto document template for industry with well grounded and accepted contents (ANSI/IEEE Standard 830 1998), Robertsons’ Volere template includes extensive project information (Robertson and Robertson 2001), and Wiegers and Leffingwell & Widrig cover business issues in their templates supported by a separate vision document (Leffingwell and Widrig 1999; Wiegers 1999).

Unfortunately, simplicity and “proven in practice” are contradicting requirements for an RD. A straightforward approach is just to provide a template that can be filled in, but in practice, simple requirements lists often prove insufficient (Kulak and Guiney 2000, p. 15). The attributes that, on the other hand, have in practice proven useful for requirements lead to far more extensive documentation (e.g. requirement source, rationale, priority, dependencies, conflicts, supporting material, test criteria, and change history). To be able to provide sufficiently wide and deep coverage of the topics addressed in RE, a documentation set of multiple document templates with appendices was developed (Nikula 2003b). The basic document templates are the RD template, the interface specification template, and the user manual template as shown in Figure 23. Since the need for an RD template has already been discussed, it is only noted that interface specification has been proposed and used by Kovitz (1998, p. 139) as a way to separate more technical requirements from the end user targeted RD, while the user manual is often suggested as a good way to document the planned system early in the development phase (Sommerville and Sawyer 1997; Kovitz 1998; Wiegers 1999; Berry et al. 2004).

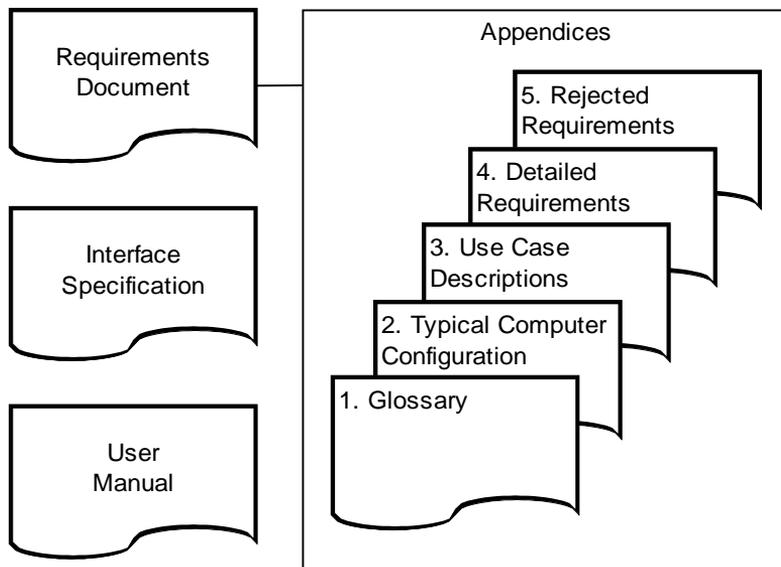


Figure 23. The BaRE requirements documentation set

The reason to have five appendices for the actual RD is to be able to address most of the foreseeable RD related problems when desired. To keep the BaRE method simple, however, it is suggested that unless the requirements engineer in the project has previous experience in documenting requirements, only the RD itself should be used. That is, documenting requirements as simple lists works well as a basic way to handle them but is bound to prove insufficient in the long run. Solving these problems requires in most cases simply more work and documentation. Thus templates that can be used to solve many of the expected problems are provided: terms can be clarified in a glossary, performance or hardware related issues might be resolved by documenting typical computer configurations, and deleted but reappearing requirements can be moved to a rejected requirements list for easy reference. Probably the most used appendices are the Use Case Descriptions and Detailed Requirements, as use cases provide a simple but efficient way to document the system usage (Kulak and Guiney 2000) and the Detailed Requirements include the attributes that have proven valuable in practice (Robertson and Robertson 1999, p. 355; Wiegers 1999, p. 274).

Altogether, the three documents and five appendices form a *two-level documentation*, where the first level refers to the requirements document and the second level to all the other documents. The general idea is that an RD is developed in every project and it is supplemented with one or more of the second level documents on a case by case basis; no projections are made about the second most important document since this is a very case dependent question. For example, most projects could benefit from a glossary; in a stand-alone software product development the Detailed Requirements –appendix could prove as the second most important document; Use Case Descriptions are useful for applications with user interactions; and User Manual may prove to be of little importance in a web-application development. Even though this two-level approach reduces the general ease of use of the method, it is important that also techniques proven in practice are available for the method users to demonstrate that many requirements-related problems can be solved.

A final point on the RD template is that the layout of the template was also considered. The layout is not a critical issue for a document, but having a well-thought layout appears important in industry and can also improve the readability of the document (Sommerville and Sawyer 1997, p. 54). Thus a standard office document layout for the BaRE templates was adopted (Finnish Standards Association SFS 2000).

6.2.3. Requirements Development Process

Synchronizing the requirements development process with the RD template makes an important contribution to the ease of use of the method. Table 18 shows the nine step requirements development process and Table 19 describes Step 1 in detail; all the nine steps are described in detail in Appendix 5. The RD template is synchronized with the development process so that Step 1 addresses only the topics in Section 1 of the RD template shown in Figure 22. In the same vein Step 2 focuses on Section 2 of the RD template. This synchronization is done when it appears feasible, and e.g. Step 3 goes back to Sections 1 & 2 to review and get a common agreement on them. Step 4 concentrates on the traditional requirements development at a detailed level. Step 5 prepares for problems by focusing on often problematic issues like constraints, assumptions, dependencies, and performance requirements. Dedicating a whole step for these issues should help in addressing them or realizing that their handling may prove insufficient. Step 6 prioritizes the requirements by allocating them to different software releases

or manual operations. Step 7 reviews the whole RD to assure that any incomplete sections get noticed and corrected before moving on with the RD. It also includes explicit tasks on developing a document overview and references section. Step 8 focuses on analyzing the RD and suggests the use of prototypes, checklists, and the CRUD (Create, Read, Update, and Delete) matrix development for different operations and entities in the software. Finally, Step 9 validates the RD with prototypes and checklists in a review meeting.

Table 18. The nine steps in the requirements development process

-
1. Establish objectives
 2. Understand context
 3. Organize knowledge
 4. Elicit requirements
 5. Prepare for problems
 6. Prioritize requirements
 7. Complete RD
 8. Analyze RD
 9. Validate RD
-

Table 19. Requirements development process Step 1: Establish objectives

RD topics	1.1 Customer problem 1.2 Business goals 1.3 Stakeholders 1.4 Product purpose 1.5 Product position statement
Participants	Initiator of the development effort Stakeholders
Techniques	Introspection Document reviews Electronic requirements Interviews Negotiation meetings
Tasks	1. Interview the initiator to get started with the task and to identify initial stakeholders. 2. Review documents and introspect to get a good idea about the objectives. 3. Interview stakeholders for deeper understanding of the task, use email when appropriate. 4. Do web searches to help develop the product position statement. 5. Arrange meeting(s) with all stakeholders to negotiate about common understanding of the objectives.

Presenting the process as consecutive steps gives it a waterfall model-like touch even though the process actually supports evolving requirements in three ways. First the two level approach to RD suggests the use of an RD template for high level requirements (i.e. business goals, context, requirements, and use case lists), while detailed requirements descriptions can be done later as appendices. Second, the process addresses the same topics in multiple steps, e.g. Step 3 reviews the requirements from Steps 1 and 2 to close the business and contextual topics before moving to specific requirements development. The third kind of support follows from the possible use of an iterative software development process. With the iterative software development process the need to address the high level topics in each iteration should be considered separately and the requirements development process should be adapted as appropriate.

As Table 19 shows, the process defines the RD topics to cover for each step and also suggests an order for developing them. Additionally participants, techniques, and tasks are provided to guide an inexperienced user through the development process. More experienced users are expected to adapt the process to their specific needs or use it in any way they see fit.

6.2.4. Method Documentation

The BaRE method documentation consists of three parts: the BaRE Guide, document templates, and the Practical RE in Short guide (Nikula 2002b). This documentation set also forms the method training material in the present study. The BaRE Guide is the principle method documentation intended to provide only information that is needed to be able to use the method in practice, and the templates should help to get started quickly. The Practical RE in Short - guide provides a wider perspective on RE by offering interested readers basic information about RE and links to alternative practices to those included in the BaRE method. The table of contents of the BaRE Guide is shown in Figure 24.

The first chapters in the BaRE Guide provide an *introduction to RE and the method* in a practical and concise way. A *quick guide on the method use* provides an overview on how to start using the method, and the *RD template* chapter explains the key issues in using the template. The *requirements development* chapter describes the requirements development and evaluation processes and provides a short description on each of the selected techniques with references to more detailed descriptions. The *requirements management* chapter describes respective topics on the RM area but expands them with the tool support topic. Finally, the *training* chapter suggests the key training needs for the method use. Altogether the BaRE Guide has 31 pages.

The Practical RE in Short guide has seven chapters: introduction, RE overview, RD templates, requirements development, requirements management, training, and alternative approaches to RE. The introduction to the document is just a simple overview of the document while the second chapter takes a broader view of RE based on the material presented at the beginning of this thesis. The next chapter describes the RD template survey and provides a general view on it with observations on RD templates in general. In the requirements development and management chapters, alternative processes, checklists, and practices are suggested for these activities. Additionally, the requirements management chapter includes information on RM tools as well as indications to surveys on them. The training chapter suggests some further training ideas. The last chapter on alternative approaches to RE describes methods that have

1 Introduction	5.3 Requirements Development Techniques
1.1 Practical Problem for the BaRE Method	5.3.1 Apprenticing
1.2 Design Principles	5.3.2 Checklists
1.3 The BaRE Elements	5.3.3 CRUD Matrix Development
1.4 Limitations of the Method	5.3.4 Define System Boundaries
1.5 How to Use This Guide	5.3.5 Define the System's Operating Env.
1.6 Few Good Books on RE	5.3.6 Diagram Development
2 BaRE Method Overview	5.3.7 Document Studies
3 Quick Guide on Using the BaRE Method	5.3.8 Electronic requirements
4 Requirements Document Template	5.3.9 Interviews
5 Requirements Development	5.3.10 Introspection
5.1 Requirements Development Process	5.3.11 Meetings
5.1.1 Process Overview	5.3.12 Prototyping
5.1.2 Step 1: Establish objectives	5.3.13 Reading
5.1.3 Step 2: Understand context	5.3.14 Reuse Requirements
5.1.4 Step 3: Organize knowledge	5.3.15 Reviews
5.1.5 Step 4: Elicit requirements	5.3.16 Scenario Development
5.1.6 Step 5: Prepare for problems	5.3.17 Two-Level Approach to RD
5.1.7 Step 6: Prioritize requirements	5.3.18 Write User Manual
5.1.8 Step 7: Complete RD	6 Requirements Management
5.1.9 Step 8: Analyze RD	6.1 Change Request Process
5.1.10 Step 9: Validate RD	6.2 Requirements Management Techniques
5.2 Evaluation Process and Checklists	6.2.1 Baselining
5.2.1 Evaluation Process	6.2.2 Change Management
5.2.2 RD Pre-Review Checklist	6.2.3 Version Identification
5.2.3 Requirements Pre-Review Checklist	6.2.4 Maintain Change History
5.2.4 Requirements Analysis Checklist	6.2.5 Tool Support
5.2.5 Requirements Validation Checklist	7 Training
	Glossary
	References

Figure 24. Table of contents of the BaRE Guide

been proposed for RE but are not included in the BaRE method. Section 6.4 of this thesis summarizes these alternative approaches and other practices from the Practical RE in Short guide. The guide includes numerous literature references for interested readers. Altogether the Practical RE in Short guide has 57 pages.

This thesis contains six appendices including examples of the BaRE documentation. Appendix 1 presents the RD topics survey that was used in defining the RD template topics for the method; it is provided also as Appendix 1 to the RE in Short guide. Appendix 4 includes the BaRE requirements document template as it is included in the method and Appendix 5 includes the requirements development process from the BaRE Guide. Appendix 6 summarizes the RE

practices and techniques discussed in the RE in Short guide as alternatives to the included practices. Finally, Appendixes 7 and 8 show the change request templates suggested in the BaRE method documentation in action; Appendix 7 includes a summary of the change suggestions to the BaRE method version 1.0 and Appendix 8 shows some change suggestions on a detailed level. This two-level documentation of change suggestions is similar to documenting requirements primarily in the requirements document and when seen important, individual requirements can be documented in detail in a Detailed Requirements appendix to the actual document.

6.2.5. Use of the Method

This section provides brief instructions on adopting the BaRE method in an organization, adapting the method to project specific needs, and how to actually use the method in a project. To start with, instructions on how to start an RE improvement effort with the BaRE method in an organization will be given.

The first step in initiating an RE practice improvement effort is to assign the role of a requirements engineer to some person. This role assignment has two purposes: first it makes this person responsible for learning RE and collecting information on it and second it gives him/her the right to invest both time and money on these tasks. Thus, in the long term, this principal requirements engineer should develop an organizational knowledge-base on RE – books, articles, templates, tools, etc., in addition to his/her own experiences – that others in the organization can utilize when needed. Another key task for this requirements engineer is to lead the adoption of new methods in the organization and help in evaluating their fit to specific projects. In case method adaptation is needed, this should also be assisted by the principle requirements engineer both by building on his/her knowledge in RE and increasing his/her knowledge with this new adaptation experience.

The adoption of the BaRE method in an organization should start with a review of the BaRE Guide to assess the fit of the method to a specific problem domain and organizational terminology. Due to the domain specificity of BaRE, the fit with the problem domain should be fairly good from the beginning. In a case where the method seems to require extensive modifications it should be considered whether to continue with BaRE or to look for alternatives. There is nothing to inhibit even major modifications to BaRE, but such actions should be left for experienced requirements engineers since they go beyond the readiness-to-use characteristic. If, on the other hand, the method appears to match the problem domain well, the process should continue with a terminology comparison. The fact that the RE terminology is far from a standardized one makes it important to prepare even for major discrepancies in the terms. After these two initial comparisons and possible adaptations, the BaRE method is ready for project level use in the organization.

For each project, the organization should first evaluate how BaRE fits its specific needs. This comparison is done in two phases – firstly, the problem domains of the project and the method are considered, and then the need for project specific adaptation is assessed. The need for another problem domain comparison is due to the possibility that the organization may run very different kinds of projects (e.g., time critical, small, large, etc.) in very different kinds of problem domains (e.g., real time, life critical, and business information system domains). The project specific needs are limited to fairly small adaptations of the method. The most common

tasks should include the addition of an attribute or two to the requirements (i.e., one or two columns in the RD template) or the use of the detailed requirements appendix due to multiple attributes for each requirement, the decision to use other appendices, the possible elimination of RD topics or the addition of new ones. Since the current RD templates and the requirements development process are synchronized, adding or deleting RD topics leads to a need to modify also the requirements development process. The elimination of RD topics can be done in two ways – either by deleting the respective section from the RD or adding some specific tag in the heading. One common way to tag the unused sections is to add “N/A” (not applicable) to the heading. This approach should be used if the organization has adopted a standard RD template to be able to identify omitted topics from the RD more easily.

The actual use of the method is quite straightforward since the processes are synchronized with the documents they are intended to produce. They also provide lists on tasks to complete, people to involve, and techniques to use for each step in a prescriptive manner for those that need it. Thus the key steps in adopting XP can be modified to suit requirements engineers starting to improve RE practices with the BaRE method (adapted from Cockburn (2002, p. 17)):

1. Do everything as written in the BaRE Guide.
2. After having done that, experiment with variations in the rules.
3. Eventually, do not care if you are doing BaRE or not.

6.3. Limitations of the Method

A fundamental assumption concerning the use of the method is that the project requirements exhibit some degree of stability after a proper development phase, as daily changing requirements can hardly be managed feasibly in a written format. The actual limitations of the method are discussed in three parts: application domain constraints, level of expertise, and individual decisions about inclusion and exclusion.

The primary constraint for the method proposed in the present study is that it is designed for small projects developing administrative and business applications. The small project size means that one requirements engineer is sufficient for the project, which in general means projects lasting from six to twelve months, having less than half a dozen workers, and a budget below 100 kEUR. This is not a tight limit and does not exclude co-operation, but it makes one person responsible for developing and managing the requirements. This limitation eliminates the problems caused by multiple persons working parallel on the same project as far as possible. Specifically, the expected issues to be avoided include inconsistent and contradicting requirements, requirements omissions, simultaneous working on the same requirements, and a need for a complex RM tool. It is clear that these questions can be solved, but due to the simple application domain their likelihood is low and addressing them explicitly unnecessary.

Considering the three levels of users by their expertise, this method is clearly targeted for the initial level people who need one procedure that works. The people on the second level want to learn alternative ways of working, different procedures, and techniques to find out the situations to which these artifacts are best suited and when they break down. For this group of users, the method provides some support through the techniques and processes it includes, even though the most support is found in the training material describing techniques not included in the basic requirements development and management components. The people on the third level of

expertise may also learn something from the method, but in general it is of marginal interest to them since they are unlikely to follow any particular procedure anyway.

The first guideline in deciding the individual topics to be included the BaRE method has been to focus on basic RE. Consequently, an explicit linkage to design or other software development activities has not been designed. Due to practical project limitations, no automated tool support for RM could be implemented, but document templates are provided as first aid for the method. On the technique side, the method suggests use of informal and semi-formal techniques, but there are no restrictions for using formal methods; they are considered as techniques and the user makes the final decisions about their use. Requirements tracing has been excluded from the basic practices, and the handling of prioritization and reviews are superficial since little can be done about these issues before the requirements are documented and managed properly.

6.4. Alternatives and Extensions to the BaRE Method

There are a number of alternative approaches and practices to RE that are not included in the BaRE method. In this section, some commonly noted alternatives to and omissions from the method will be noted. The alternative approaches to RE will be the first step, followed by the more detailed practices.

Contextual design (Beyer and Holtzblatt 1998) has recently received considerable attention in the area of user centered design. Beyer and Holtzblatt state that contextual design supports finding out how people work in order to be able to discover an optimal redesign of the work practices in large and complex projects, but it has also been used successfully on small projects. One key aspect of contextual design is that it provides explicit steps and deliverables from initial discovery through system specification. The six major parts of this approach are the following:

- **Contextual Inquiry.** This part focuses on understanding the customer's needs, desires, and approaches to work; this activity is based on talking to the customers while they work.
- **Work modeling.** A complex and unfamiliar work domain may prove hard to understand for designers, so work models – diagrams – are built to create a concrete representation of people's work.
- **Consolidation.** Data from different customers is brought together and considering the multiple customer diagrams, a single picture of the work models is developed.
- **Work redesign.** Based on the consolidated data, conversions are conducted with the customers, issues are raised, and a vision is developed of the new way of working.
- **User environment design.** Each part of the system is shown in the user environment design, how it supports the user's work, exactly what function is available in that part, and how it connects with other parts of the system. However, it does not tie the structure to any user interface.
- **Mock-up and test with customers.** The system design is tested with rough paper prototypes to get feedback on the structure and user interface.

The **Soft Systems Methodology (SSM)** is among the best known problem-solving methodologies in the IS field (Checkland 1981; Checkland and Scholes 1990). The seven-stage SSM approach (Figure 25) has raised a lot of interest and it has been applied to many purposes, e.g. strategic planning for business improvement (Bustard and Wilkie 1999), supporting the RE phase (Paton 1992), and prioritizing requirements (Dearden and Howard 1998). However, SSM has also been criticized. The main problem of SSM is its limitation to the problem-solving part, which leads to a need for other methods to be able to define requirements in sufficient detail for implementation (Probert 1999). This also means that the methodology does not provide any requirements related templates, clear practices or processes.

SSM is included in this part of the thesis to provide another view to complement the straightforward approach adopted in the BaRE method. As a matter of a fact, Checkland has detached himself from the seven-stage model shown in Figure 25 as too simple a model. The general nature of the SMM is well described by two quotes (Checkland and Scholes 1990, p. 298-299):

“the white lie is that SSM is a seven-stage problem-solving methodology applicable to problems of a certain kind, namely messy, ill-structured ones. We can forgive the pedagogues’ simplification; after all, the true complexity of the real world *has* to be simplified for classroom consumption. But in the real world outside classrooms, if we want seriously to bring about improvements, we need to work with the richer account of SSM given above.”

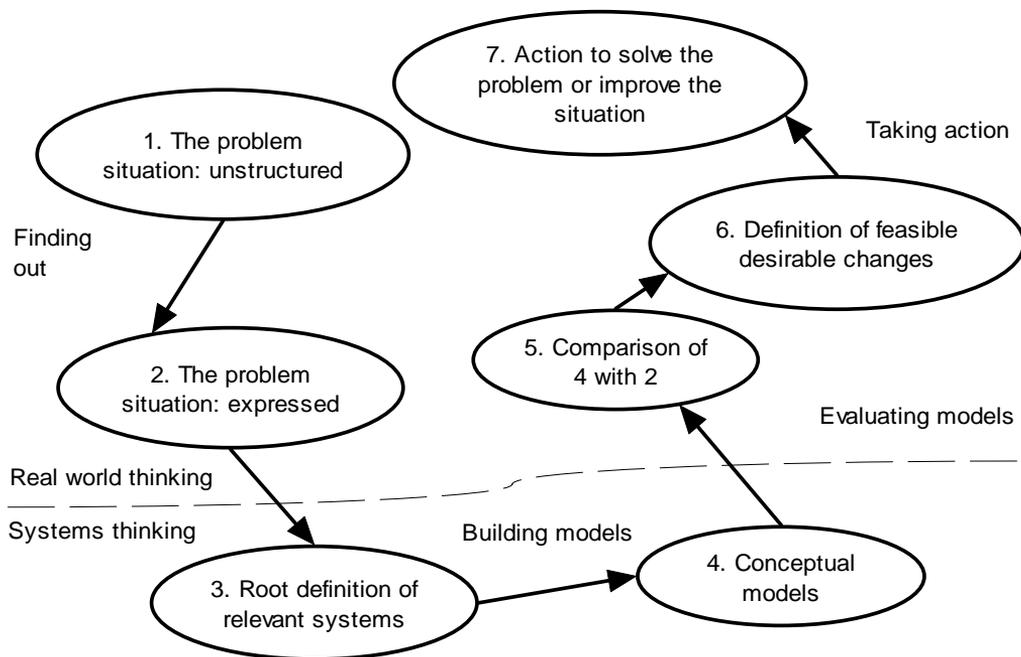


Figure 25. Checkland’s seven-stage Soft Systems Methodology (Bustard and Wilkie 1999, p. 2)

“When people hear for the first time about SSM, or systems engineering, or cognitive mapping, or any such methodology, they naturally ask: Is it any good? Does it work? And they are very frustrated when they develop an understanding of the nature of methodology sufficient to realize that the question is unanswerable, that methodology is in fact *undecidable*.”

Another alternative approach to RE is the **goal approach**. The BaRE RD template includes business goals but since the goals are handled in a superficial way, key references to related research efforts are provided. According to van Lamsweerde (2000a), the first goal centric methods were suggested in the late 1970's. However, probably the best known goal centric method – the Goal-Question-Metric (GQM) paradigm – was presented in late the 1980's by Basili (1989). After that a number of other methods utilizing goals have been proposed, (e.g. Antón 1996; Cockburn 1997; Lee and Xue 1999; Antón et al. 2001).

The purpose of **viewpoint (VP) oriented techniques** is in general to help identify different stakeholders and manage requirements collected from them. Since BaRE is a basic method, the RD template includes sections for stakeholders and different user classes but there is no explicit support for identifying and managing the possibly conflicting requirements from the different viewpoints. The basic idea of viewpoints is presented in Figure 26, which shows the incomplete views that different viewpoints have on both the problem and the system. The principles of viewpoints have been presented in multiple sources, (e.g. Sommerville and Sawyer 1997;

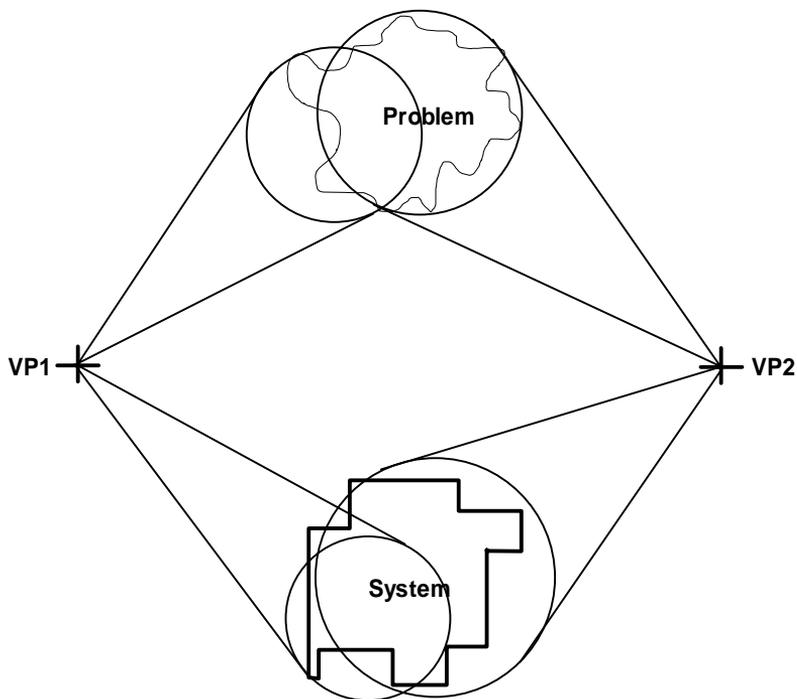


Figure 26. Viewpoints on the problem and the system (Sommerville and Sawyer 1997, p. 361)

Sommerville et al. 1999) in addition to an earlier paper presenting also the VORD method (Kotonya and Sommerville 1996). Easterbrook and Chechik (2001), on the other hand, propose a framework for reasoning over inconsistent viewpoints. As the last point, it is good to notice that there are also research efforts that suggest that there are some limits to the utility of viewpoints in requirements modeling (Menzies et al. 1999).

Problem frames approach RE from the patterns viewpoint and introduce patterns into the problem domain. The problem frames were first published in the book by Jackson (1995) with only a few pages. However, in his later book devoted to problem frames Jackson (2001) refines the idea to some extent and presents a comprehensive treatment of the topic. The key problem frames are required behavior, commanded behavior, information display, simple workpieces, and transformation frames (Jackson 2001). These five frames do not represent all the problem frames in the world, but Jackson believes that the most common problems can be presented with a few frame flavors and variants complemented with composite frames. The problem frames have much in common with design patterns, but according to Jackson they are still two different things since they focus on different areas: design and analysis. The key idea stays the same, though – once a familiar pattern in the problem domain is identified the prior knowledge in solving the problem can be utilized. Despite the benefits of this approach, the problem frames were not included in the BaRE method since they are new to most potential method adopters and this was considered as a potential inhibitor for the method adoption (McPhee and Eberlein 2002).

As a final approach to RE, the **quality function deployment (QFD)** will be discussed (Liu 2000). QFD was developed in the 1960's in Japan to expand and implement the quality techniques in shipyard industry taught by W. Edwards Deming and others. After that these techniques have been applied in many industries worldwide including automobile, electronics, and software industries. The key steps in the software QFD process are the following: 1. customer requirements, 2. technical product specification, 3. the correlation matrix, 4. customer requirements priorities, and 5. technical product specification priorities (Haag et al. 1996). Even though the use of QFD is reported to be increasing (Haag et al. 1996) and having clear benefits (Krogstie 1999), it was not considered familiar and simple enough for inclusion in the BaRE method.

Moving from the approaches to practices, it is noticed that the multitude of choices remains the same. For example, the key references in good RE practices (Sommerville and Sawyer 1997; Wiegers 1999) suggest 66 and 45 practices respectively. Since the BaRE method was planned to be a basic RE method, most of these practices are not suggested in the BaRE Guide. However, the supplementing Practical RE in Short guide lists the rest of the good practices from the above-mentioned books; they are also summarized in this thesis as Appendix 6. Thus an interested practitioner can find other useful techniques from the BaRE method documentation. In the same vein, the RD template survey done as part of the BaRE RD template development resulted in a vast number of potential RD topics that are summarized in the BaRE method documentation and Appendix 1 of this thesis. A reader interested in writing requirements documents might also find an ambiguity handbook by Berry et al. (2003) interesting reading since it focuses on linguistic sources of ambiguity and includes over 400 illuminating examples on the topic.

This section on alternatives and extensions to the BaRE method is closed with initial results from a larger research effort to increase the understanding of how experts use different

requirements elicitation techniques and what factors they consider in the technique selection process (e.g., Hickey and Davis 2003a; Hickey and Davis 2003b). As the conclusions of their interviews with some of the world's most experienced analysts, Hickey and Davis (2003a) report that for each elicitation technique there seems to be both major drivers that lead experts to consider each technique and anomalies that may cause experts to alter their primary choice. It is also reported that for each elicitation technique there seems to be a set of basic analyst skills that must be present for the technique to be effective, *prerequisite skills*, and a set of additional skills that are not universally needed but that come into play during the technique's execution without pre-knowledge, *success enhancers*. Further, as a more specific conclusion, Hickey and Davis report that almost all experts use models which seem to help in fully comprehending a situation and in communicating with stakeholders. Thus this research effort may provide important practical insights into how to select elicitation techniques in the future.

6.5. Summary

This chapter started by presenting the five attributes Avison and Fitzgerald (2003) expect from a method: philosophy, phases, techniques, tools, and training. Considering the specific circumstances of the present study, the BaRE method components were summarized as requirements document templates, requirements development practices, requirements management practices, tool support for requirements management, and training. Each of the method components was then described in more detail.

The rest of this chapter considered the constructed method from different viewpoints. First, the limitations of the method were presented by first noting that documenting requirements may prove impractical in a project with very many changes and, thus, some stability should be born in the project where the BaRE method is applied. The other main limitations of the method included domain specificity, focus on low maturity organizations and people, focus on RE only, and lack of automated tool support that would follow the same principles as the method itself. As the last point in this chapter, alternatives and extensions for the method were presented including also higher level approaches to problem solving and specific techniques to assure that all requirements are treated consistently. The reason for including such a diverse collection of discrepant approaches was to provide a starting point for more extensive search for alternative and complementary approaches in the early development phases.

7. LITERATURE BASED EVALUATION

This chapter focuses on the literature based evaluation of the BaRE method. The method is evaluated with the REAIMS Top Ten assessment (Section 7.1), the key issues in RE tool and technique selection (Section 7.2), the software project success factors (Section 7.3), and the software project risk factors (Section 7.4). As the last evaluation in this chapter a survey of other RE methods is reported and the BaRE method is compared with three methods that appeared most similar to it (Section 7.5). The section is closed with a discussion on the literature based evaluation including its importance for the ease of adoption idea in general (Section 7.6). Notice that the BaRE method has also been estimated in the light of the Three Dimensions of RE framework outlined in Section 3.2.1. However, as this evaluation provided little further insight, it is not repeated here; an interested reader can find the evaluation results elsewhere (Nikula 2002a; Nikula 2003a). Even though this chapter does not address any research question as such, it was found important for two reasons. First, it provided estimates on the method potential before any empirical evaluations had been conducted with considerably less resources, and second, it reports on comparisons with related work which is in general found important in research, for example, to show the novelty of the conducted work.

7.1. REAIMS Top Ten Assessment

A proper REAIMS assessment requires knowledge on both the guidelines and their use in practice, so it is not optimal for a desktop evaluation. However, evaluating the way the method addresses each guideline was possible, and the BaRE method was evaluated with respect to the top ten guidelines. Since there was an awareness of the guidelines already in the design phase of the method, it was natural that all the guidelines were addressed to some extent by the method (Table 20). A more interesting observation was that addressing some of the top ten guidelines appeared difficult to support by design and some seemed unnecessary for small projects in the administrative and business application domain. Thus the guidelines were reorganized into two groups where the first group is formed of five guidelines that can be addressed at the method level and the second group is formed of the five other guidelines. The guidelines in the second group can only be supported somehow in the method since their actual implementation depends on the person doing the work. To emphasize the difference between these two groups, the first five guidelines are called *infrastructure* and the last five guidelines *working practices*; the working practices are identified in with the asterisk (*) symbol.

7.2. Key Issues in RE Tool and Technique Selection

McPhee and Eberlein (2002) identified the top ten techniques for RE from three different viewpoints – familiarity, usefulness for time-to-market (TTM), and usefulness for non-time-to-market projects (Table 14, p. 70). Since the BaRE method does not claim to include aspects supporting TTM projects, a match with the techniques most useful for Non-TTM projects is considered first. As Table 21 shows, seven of the useful techniques are suggested in the BaRE method, two techniques are suggested as extensions to the basic practices in the Practical RE in Short Guide, and this guide also explains the viewpoint oriented techniques not directly advocated anywhere in the method documentation. Thus the BaRE method addresses these techniques quite well.

Table 20. The REAIMS top ten guidelines and the BaRE techniques to address them; working practices are identified with the asterisk (*) symbol

REAIMS Top Ten Guideline	BaRE Techniques to Address the Guideline
Define a standard document structure	Adopted, an RD template provided
Define standard templates for requirements descriptions	Adopted, both RD and second level documents suggest standard attributes for requirements
Uniquely identify each requirement	Adopted
Define validation checklists	Adopted, initial checklists provided
Define policies for requirements management	Applicability to small projects questionable, RM practices suggested instead
Make the document easy to change*	This guideline is of marginal importance due to small projects and short documents Supported by multiple logically grouped documents Supported by instructions
Use language simply, consistently, and concisely*	Supported by use of tables and example requirements
Organize formal requirements inspections*	This guideline appears too advanced for small administrative and business application projects, informal reviews suggested instead
Use checklists for requirements analysis*	Adopted, initial checklists provided
Plan for conflicts and conflict resolution*	Supported by active user collaboration and suggesting a negotiation meeting structure (Nikula 2002b, The BaRE Guide, p. 23)

The most useful techniques for TTM projects include only three techniques that were not within the top ten techniques for a Non-TTM project: requirements prioritization, evolutionary prototyping, and requirements reuse. All these techniques are suggested in the BaRE method, even though prototyping does not differentiate evolutionary and throw-away prototypes but suggests only different types of throw-away prototypes.

Considering the RE techniques from the point of view of familiarity, only informal modeling and throw-away prototyping techniques were not within the useful techniques for Non-TTM projects. As was just noted, the BaRE method does suggest different kinds of throw-away prototypes and since the method suggests natural language documentation with tables and semi-formal language diagrams as defined by Pohl (1994), it can be claimed that the BaRE method supports informal and semi-formal representations or modeling. Thus there is fair evidence that the BaRE method addresses the most useful and familiar techniques for RE as suggested by McPhee and Eberlein (2002) quite well.

Table 21. The ten most useful RE techniques for a Non-TTM project (McPhee and Eberlein 2002) and the BaRE method approach to them

Rank	Usefulness (Non-TTM)	BaRE Approach
1	Requirements Change Management	Suggested
2	Semi-Formal Modeling	Suggested
3	Requirements Reviews	Suggested
4	Scenarios/Use Cases	Suggested
5	Requirements Checklists	Suggested
6	Requirements Testing	Suggested as an extension
7	Requirements Tracing	Suggested as an extension
8	Viewpoint-Oriented Techniques	Explained in Practical RE in Short -Guide
9	Interviews	Suggested
10	Designer as Apprentice	Suggested

7.3. Software Project Success Factors

The study by Hofmann and Lehner (2001) came up with ten RE practices that the authors claim to have acted as software project success factors in the projects they studied (Table 6, p. 42). These best RE practices and the BaRE approach to them is shown in Table 22.

Table 22. The ten best RE practices (Hofmann and Lehner 2001) and the BaRE approach to them

Best Practice	BaRE Approach
Involve customers and users throughout RE	Suggested; <i>throughout</i> not emphasized
Identify and consult all likely sources of requirements	Suggested; <i>all likely sources</i> not emphasized
Assign skilled project managers and team members to RE activities	Out of RE scope, a project management task; competence issue acknowledged
Allocate 15 to 30 percent of total project effort to RE activities	Out of RE scope, a project management task
Provide specification templates and examples	Provided; a full RD example missing
Maintain good relationships among stakeholders	Out of RE scope; no single practice, technique etc. can achieve this
Prioritize requirements	Suggested
Develop complementary models together with prototypes	Basically; some models should be developed even if many would not be possible
Maintain a traceability matrix	Tracing suggested as an extension
Use peer reviews, scenarios, and walk-throughs to validate and verify requirements	Suggested

The BaRE method suggests two of these practices just as they are and three in principal; the Hofmann and Lehner practices include some emphases that are not acknowledged in BaRE – for example, customers and users should be involved in RE but *throughout* is not emphasized in BaRE, and on the other hand, BaRE advocates identifying and consulting different stakeholders but does not suggest this should cover *all likely sources*. BaRE also suggests prototyping and developing some models (or diagrams or *descriptions* as defined by Jackson (1995, p. 121)) but developing *complementary* models appears unrealistic in organizations that are mostly learning to document. There is one practice, maintaining a traceability matrix, that is suggested in the Practical RE in Short guide in a more general way as *tracing requirements*. Finally, three of the reported best RE practices appear to be out of RE scope: assigning people, allocating project effort, and maintaining good relationships among stakeholders. These practices are clearly related to RE, affect the outcome of RE, and can/should be done as a part of it. However, these issues are better addressed as project management activities and possibly as part of training than as single RE techniques, practices, or actions. Overall, the BaRE method aligns fairly well with the ten best RE practices suggested by Hofmann and Lehner (2001) in the Table 22.

7.4. Software Project Risk Factors

The seven key requirements-related risk factors for software projects summarized in Table 7 (p. 42) are listed in Table 23 together with the ways the BaRE method tries to mitigate them. In general, BaRE tries to alleviate the likelihood of these risks and, on the other hand, tries to make the presence of these risks apparent so that appropriate actions can be taken. Based on the data in Table 23, the BaRE method could well prove helpful in software project risk mitigation.

7.5. Evaluation against other Requirements Engineering Methods

Evaluating a method with discrepant frameworks provides some insight about its characteristics but comparisons with other methods are required to estimate the novelty of the method. Thus the BaRE method was evaluated against other methods to assure the novelty of the selected approach. This evaluation was conducted in two steps; first, the way different methods addressed some of the key components of the BaRE method were considered, and second, adaptability, readiness-to-use, and ease of use of the remaining methods were considered more closely. These three characteristics were selected due to their importance from the point of view of the ease of adoption.

It has been estimated that there are over a thousand brand name methods for systems development (Fitzgerald 1998, p. 318) but the number of RE methods is naturally much smaller, which made this survey a realistic task. The literature survey covered 14 RE books, one book discussing the Unified Software Development Process (Jacobson et al. 1998), and the Small Project Roadmap for the Rational Unified Process (RUP) (Rational Software Corporation 2002). The Jacobson et al. (1998) book provided a reference point for RE books, while the Small Project Roadmap presented an example of a process tailored from RUP. No attempt was made to cover general (information) systems methods that in many cases cover also RE, but RUP served as the only method covering the whole software development and life-cycle. The Software Engineering Institute's Software Capability Maturity Model (CMM) in the review was also considered, but since it addresses RE merely as two requirements management practices (Software Engineering Institute 1995, p. 126) it did not provide the clear practical level help that was sought, and thus CMM was excluded from further study.

Table 23. The requirements-related risk factors and the ways BaRE tries to mitigate them

Risk factor	BaRE techniques to mitigate the risk factors
Misunderstanding the requirements	Documenting requirements Interacting with the stakeholders Reviewing RD and requirements
Lack of adequate user involvement	Using techniques involving users
Failure to manage end user expectations	Interacting with users Documenting requirements Prioritizing requirements Validating requirements with prototypes
Changing scope/objections	Documenting business goals, context, and requirements Requirements management: baselining RD and requirements, change requests
Lack of frozen requirements	See the techniques to mitigate <i>Changing scope</i> -risk Documenting requirements Use of second level RD for detailed descriptions Change management
Conflict between user departments	Documenting requirements Meetings with all stakeholders Reviewing RD and requirements
Incomplete requirements and specifications	Documenting requirements Use of second level RD for detailed descriptions Requirements analysis techniques Reviewing RD and requirements

Table 24 summarizes the results of the survey in six areas of interest which were derived from the basic components of the BaRE method (Section 6.1, p. 98). The requirements document template was the first component of interest and it was supplemented with the requirements development and management practices covering both techniques and processes in both the areas. The requirements development techniques consisted of elicitation, analysis, specification, and validation techniques, while RM techniques consisted of change control, version control both for individual requirements and requirements documents, requirements tracing, and requirements statuses (Wiegiers 1999, p. 38). The development process was to describe how to use techniques in a systematic way to produce requirements, while the RM process was to explain how to manage changes in requirements. The expected level of detail was defined to provide an accurate enough description for a reader to test the suggested template, techniques, and processes in practice. The last component of interest was the packaging of the solution as a method. Full treatment of the subject is marked with the character ‘●’ in the respective cell in the table while partial treatment is marked with the character ‘○’ and lack of treatment is indicated with a dash.

The comparison presented in Table 24 leads to two interesting observations. First, only four sources cover both requirement development and management areas (Kotonya and Sommerville

1998; Leffingwell and Widrig 1999; Wiegers 1999; Rational Software Corporation 2002), and second, only three of these sources suggest some kind of package that can be called a method. Since packaging of the solution was found important for the ease of adoption, Wiegers' book was excluded from closer consideration. The rest of this section discusses the three methods closer from the point of view of readiness-to-use and ease of use in small administrative and business application projects.

Leffingwell and Widrig (1999) is an introductory book on RE and it provides a comprehensive approach to RE. The book also includes a prescription for an initial process – a recipe (p. 94) – that has a number of assumptions limiting its applicability, e.g., the project size is estimated to be 10-30 team members and the application is a single, stand-alone one without contractual requirements for documentation. The followers of the provided prescription are expected to have read and understood the Leffingwell and Widrig book or have otherwise respective knowledge in RE. The eight steps of this process are summarized in Table 25; the actual prescription in the book lists four to eight tasks for each step covering and referring to the key techniques presented earlier in the book.

Table 24. Items reviewed for method development. The character '●' indicates full treatment of the subject, the character '○' partial treatment, and the dash lack of treatment

Reference	Doc	Development		Management		Method
	Templ	Techn	Process	Techn	Process	
(Davis 1993)	●	●	-	-	-	-
(Graham 1998)	-	-	●	-	-	-
(Hooks and Farry 2000)	-	-	●	○	●	-
(Jacobson et al. 1998)	-	-	●	-	-	-
(Kotonya and Sommerville 1998)	●	●	●	●	●	○
(Kovitz 1998)	●	-	-	-	-	-
(Kulak and Guiney 2000)	●	●	●	-	-	-
(Lauesen 2002)	●	●	-	-	●	-
(Leffingwell and Widrig 1999)	●	●	●	●	●	●
(Maciaszek 2001)	●	●	-	○	○	-
(McGraw and Harbison 1997)	-	●	●	-	-	-
RUP Small Project Roadmap (Rational Software Corporation 2002)	●	●	●	○	●	●
(Robertson and Robertson 1999)	●	●	●	○	-	-
(Sommerville and Sawyer 1997)	-	●	●	○	-	-
(Wiegers 1999)	●	●	●	●	●	-
(Young 2001)	-	-	●	-	-	-

Table 25. The eight steps in the Leffingwell and Widrig recipe to get started with RE (Leffingwell and Widrig 1999, p. 396)

-
- Step 1: Understand the Problem Being Solved
 - Step 2: Understand User Needs
 - Step 3: Define the System
 - Step 4: Continuously Manage Scope and Manage Change
 - Step 5: Refine the System Definition
 - Step 6: Build the Right System
 - Step 7: Manage the Requirements Process
 - Step 8: Congratulations! You've Shipped a Product!
-

From the small administrative and business application point of view, the recipe supported by the book appears ready-to-use. Namely, all the basic elements are suggested: document templates are provided, techniques and processes are outlined both for requirements development and management, training material is provided in the form of the book including examples – the book even suggests a tool to support RE. A case study is developed based on the techniques in the book providing a good reference point and practical example of the suggested techniques. The case study describes a home lighting automation system project including hardware and software parts and a 15 person team. Without more specific knowledge of the need for the method, the Leffingwell and Widrig approach seems reasonable. The ease of use of the recipe can be questioned due to the 491 pages in the book with the recipe being suggested only on page 396. However, the book provides a thorough introduction to RE with many examples and templates, so the size of the book is understandable. In short, the book provides good support for RE improvement efforts. However, it can be speculated whether the ordering of the topics and the extensive technique coverage with extensive templates build an invisible barrier for adoption in projects having only few people. The adaptability is not considered as such in the Leffingwell and Widrig (1999) book since the recipe was introduced as a way to integrate the suggested techniques in a prescriptive way.

Kotonya and Sommerville (1998) is another introductory book on RE and it describes the VORD method as an example of interactive system specification methods. VORD stands for Viewpoint Oriented Requirements Definition and it is consequently focused only on the requirements development phase, but the introduction part provides also a proper handling of requirements management issues. VORD was decided to be given a closer look as it was the only method that was designed specifically for requirements development. The primary source of information for VORD was the Kotonya and Sommerville book (1998); a VORDTool was also found to support the viewpoint oriented RE (Kotonya and Sommerville 1997); and two articles describe VORD and experiences from it in practice (Kotonya and Sommerville 1996; Kotonya 1999).

The key concerns of interactive systems include user interface, user classes, other systems, indirect system concerns, and quality of service (Kotonya and Sommerville 1998, p. 215). Consequently, the 30-page description of the VORD method includes excerpts from an automated teller machine specification to demonstrate how VORD suits for applications where reliability, performance, and organization play an important role; the example includes 15 different diagrams and tables. Since no actual method documentation or guide could be found to

supplement the research papers and the book, neither readiness-to-use nor ease of use of the method could be determined. A fundamental issue with VORD is that it is a viewpoint-oriented technique and almost all the seen document fragments embody this characteristic. Thus moving away from the viewpoint approach would require extensive adaptations to the suggested documentation, which makes the adaptability and readiness-to-use questionable. As a whole, VORD does not seem to be a basic method that would provide an easy start in doing RE.

The RUP Small Project Roadmap suggests an impressive set of artifacts to start carrying out RE in small projects (Rational Software Corporation 2002). A lot of effort appears to have been invested into making the product adaptable, and there are numerous requirements document templates and extensive explanations on customizing RUP for the situation at hand. As a matter of fact, the RUP Small Project Roadmap is a set of instructions on how to modify RUP to fit small projects. RUP itself is a set of tools and techniques targeted at large and complex projects (Jacobson et al. 1998). Due to its adaptability, RUP is usable in a variety of situations and domains, and there is no point in talking about domain specificity with it; it has also been claimed that RUP should always be tailored to the project at hand (Cockburn 2002, p.199).

The general RUP approach with use cases and an iterative process model should fit administrative and business applications well. However, since the primary target of RUP is large projects, it is unlikely to be very ready-to-use for small projects without adaptations. The basic approach to tailoring the process does not appear very simple: “Most of the RUP activities and artifacts are needed on a small project – the differences are more in terms of artifact formats and the level of formality, detail, and effort applied to each activity” (Rational Software Corporation 2002). The instructions on the tailoring process say that one should first understand the RUP basics, such as process essentials, tailoring the process, and best practices for developing software before domain specific process adaptations are developed. Since RUP is a software development process, using it for RE only would lead to a situation where a majority of the process should be discarded. In short, the readiness-to-use of the RUP for small projects appears very limited and the RUP cannot be considered a promising approach from the basic RE point of view.

The BaRE method differs from these three methods in significant ways. In comparison with the Leffingwell and Widrig (1999) recipe, the BaRE method does not present major differences in domain specificity or adaptability, but integration between the RD template and the requirements development process is much tighter in the BaRE method. For example, in addition to the tasks to complete, the BaRE development process suggests the RD topics to address in each step, people to involve, and techniques to use. Since the situation is the same with RM, it is claimed that BaRE takes a step further in the readiness-to-use and, consequently, it provides an easier way to start doing RE in practice than the Leffingwell and Widrig approach. Comparison with VORD was limited to the requirements development phase but it revealed major differences between the methods. VORD was developed to provide a close integration with the viewpoint approach, which led to serious limitations in the adaptability of the method. The number of suggested documents, diagrams, and tables implies that VORD is targeted for complete and unambiguous specification which, on the other hand, suggests that it is not very easy to use. Finally, no real guide or instructions for the use of the method could be found. Thus BaRE appears to be more adaptable, easier-to-use, and readier-to-use than VORD in the selected application domain. Since the RUP Small Project Roadmap requires understanding of the basic RUP, BaRE was compared with RUP as such. It is clear that RUP provides much more adaptability than BaRE as, for example, it is not domain specific. The

generality of RUP, however, taxes its readiness-to-use and ease of use heavily. In practice, RUP can not be called easy to use or ready-to-use, and thus BaRE provides a much easier and faster start on doing RE in its limited domain.

Overall, the conducted survey indicates that most of the found methods did not cover the areas of interest very well. Further, none of the methods appeared very ready-to-use for the target application domain or were found to be particularly easy to use. Thus, based on the conducted survey, the constructed BaRE method appears easier to adopt than the other studied methods.

7.6. Discussion and Conclusions

This chapter has focused on the literature based evaluation of the BaRE method. The rest of this section summarizes these evaluations and provides a discussion and conclusions of the literature based evaluations.

The evaluation against the REAIMS Top Ten guidelines did not bring about any big surprises since the BaRE method was designed with those guidelines in mind. It should be noticed, though, that only half of the suggested guidelines can be implemented by a method directly (i.e., infrastructure) and the other half are inherently dependent on the person doing the work (i.e., working practices). The comparison of the BaRE method and the key issues in RE tool and technique selection was concluded by claiming that the method addresses the most useful and familiar techniques suggested in the article referred to. The third comparison with software project success factors revealed that the article referred to had a bit broader scope for practices than the BaRE method including some project management issues and, on the other hand, it also included some additional emphasis for some of the success factors. However, bearing in mind that the projects used to identify the success factors were critical business applications involving efforts of about 120 person months, the BaRE method can be claimed to compare fairly well since it addresses most of the suggested practices to some extent. Finally, in the evaluation against requirements related software project risk factors it was noted that BaRE has two objectives: first, to alleviate the likelihood of the risks, and second, to make their presence apparent. The most common technique for both of these purposes is documenting the requirements. Other suggested key risk mitigation techniques included interacting with users and reviewing requirements and requirements documents. It is important to notice, however, that risk management is a project management activity and it is addressed in BaRE only as a by-product of the adopted RE practices.

Based on the comparison with the other RE methods, the BaRE method appeared to offer a quick start to carrying out requirements development and management in a software project. The weakest part of the method was clearly the lack of tool support for RM, which was alleviated by providing document templates and, on the other hand, by referring to freeware and commercial tools readily available in the market. The finished method does not look as simple as was originally expected, but further simplification would have risked omitting important parts of RE, so in lack of specific information about the adopting project, application domain, and organization, elimination of further parts of the method was not justified. Considering that RE is one of the hardest parts of software development, simplifying the method indefinitely is hardly a realistic goal. Looking at the selected application domain, the BaRE method appeared to provide a fair compromise in limiting the method size while still achieving a reasonably wide

applicability. In short, the BaRE method is believed to have achieved its original goals fairly well and to be fairly easy to adopt.

The results of the literature based evaluation have implications on easy to adopt methods in general. Namely, failure to find truly comparable methods in the literature provides evidence for the novelty of the idea in the RE area. Considering that the constructed method did address most of the concerns reported in literature as summarized in this chapter, the method appears to bear a fair amount of potential to be useful in practice. Thus it appears justified to invest in the empirical evaluation of the constructed method in an industrial setting.

8. EMPIRICAL EVALUATION

This chapter presents the empirical evaluation of the BaRE method that was conducted as case studies in three companies. First Section 8.1 provides an introduction to the conducted empirical evaluation and then Section 8.2 describes the conducted case studies. The different contexts for change are looked at in Section 8.3 and the findings of the case studies are discussed with the conclusions in Section 8.4. This chapter is closed with a summary in Section 8.5. Individual case study descriptions supplementing this chapter are available in Appendix 2 and suggested changes to the BaRE method version 1.0 are documented in Appendix 7 to this thesis. The BaRE method documentation in this chapter refers to the BaRE Method version 1.0 (Nikula 2002b).

8.1. Introduction

In this section, research methods and the assessment tools used are described and then an overview of the conducted case studies is provided.

8.1.1. Research Methods

The empirical evaluation was conducted as case studies in three companies following the guidelines for case study research suggested by Yin (2002). The four criteria for research design quality as suggested by Yin are explained in more detail in Section 1.3.2 (p. 20), but in short, data was collected from multiple sources. All the developed requirements documents were collected from the projects completed in the case studies and other related documents when possible (e.g., submitted change requests, developed change management guides and document templates, and review records). It was also possible to obtain some archival records showing the planned and completed activities with time statistics. The conducted interviews included survey, focused, and open ended questions to collect data for specific questions. Finally, it was possible to get the SPICE RE process capability assessment results from an independent SPICE analysis conducted in the participating companies. To establish a chain of evidence, the meetings during the study were audio recorded, transcribed, and analyzed with the ATLAS.ti application (Scientific Software Development 2004). A separate diary was kept for each case study where all the contacts were recorded in brief and all the emails and documents were stored for further study in the analysis phase. The case study reports are shown in Appendix 2; these reports were reviewed and accepted by the key informants participating in the case studies.

8.1.2. Used Assessment Tools

To be able to estimate the effect people had on the outcome of the project, some basic information about the participants was gathered. First of all, the job titles, education, and experience in software development work were recorded from all the participants. The education is indicated with one of the three letters – B, B*, and M – where B stands for a bachelor's degree, B* for a bachelor level education in a master's program⁵, and M for a

⁵ B* is due to the local university education system where a Master's degree can be achieved directly without a Bachelor's degree.

master’s degree. To be able to estimate the level of knowledge people had in RE also their competences were estimated with a lightweight approach developed from Vickers et al. (2002). This assessment scheme has two obvious shortcomings: it is not documented in detail in the public domain and nothing is known about its psychometric validation. However, the approach taken in the assessment matches the approach of the present study and, on the other hand, the assessment was developed by Professor Michael A Jackson for the Praxis Critical Systems Limited, UK, where it is also being used (personal communication with Dr Andrew Vickers, Praxis Critical Systems Limited, UK).

The adapted self assessment covers general competences, competence in processes, and competence in RE techniques and specification (Table 26, adapted from Appendix 3. Competence Evaluation Form). The general and process competences were evaluated as single composite areas while the RE techniques and requirements document topics were covered as five and nine areas respectively; the numbers in parenthesis in Table 26 indicate the counts of the RE techniques and subtopics in the requirements document template suggested in the BaRE Guide. The competence in each area was estimated on five levels as shown in Table 2 (p. 30): **N**ovice, **T**rainee, **S**upervised practitioner, **P**ractitioner, and **E**xpert. Considering these levels it is clear that the BaRE method is targeted to help Novices to achieve a Trainee level by providing information for them and to support in the first few projects to move from the Trainee to Supervised practitioner level. Since changes in competences do not happen quickly, the competences were measured only once to see the level of knowledge the participating people had during the project. Notice that the first letters of the competence levels are used later on as abbreviations of the levels.

Table 26. Competence evaluation areas

Competence Area	Specific Areas
General	Overall competence in Software Engineering, Requirements Engineering, and Application Domain
Processes	Overall competence in Software Development, Requirements Development, Change Management, and Review processes
RE Techniques (From the BaRE Guide)	Elicitation (10) Analysis (5) Validation (4) Requirements management (5) Documenting (1)
Documenting Requirements (From the BaRE Requirements document template)	Business view (3) Product description (9) Application domain (2) Prioritization (4) Performance (4) Data (2) Software system attributes (8) Non-functional requirements (4) Miscellaneous (4)

In the course of the study, the company practices were assessed multiple times to track the progress in the companies. To avoid disturbing the daily routines of the small companies too much, a lightweight adaptation of the REAIMS assessment was developed, the CMM level was estimated based only on discussions about company practices, a competence evaluation method was adapted to the needs of the present study, and an infrastructure checklist was developed that could be used both to evaluate the available infrastructure for RE and how it was used in practice. Since all these measurements were lightweight adaptations, an opportunity to review the results of standard SPICE assessments (El Emam et al. 1998; ISO/IEC TR 15504-2 1998) in the participating companies was used, and the key results from them are reported here, as well. The SPICE assessments were done at the end of the project by the tSoft project and no public reports from them are available.

The Lightweight REAIMS Top Ten assessment was developed based on the experiences from using the Top Ten adaptation of the REAIMS assessment earlier in the state-of-the-practice investigation (Section 4.1, p. 73) and the literature based evaluation of the method. The lightweight adaptation differs from the REAIMS Top Ten assessment (Section 3.2.1, p. 47) in two ways: two of the guidelines were modified to less formal ones and standardized use was modified to require only documented use of the guidelines without quality assurance checks. The changes concerned the “Define policies for requirements management” and “Organize formal requirements inspections” guidelines; first the word *policies* was changed to *practices* in the former guideline and then the words *formal inspections* to *informal reviews* in the latter guideline. The Lightweight REAIMS Top Ten guidelines are summarized in Table 27.

Table 27. The Lightweight REAIMS Top Ten guidelines

Define practices for requirements management.
Define a standard document structure.
Define unique identifier for each requirement.
Define standard templates for requirements description.
Define validation checklists.
Organize informal requirements reviews.
Use language simply, consistently, and concisely.
Make the document easy to change.
Use checklists for requirements analysis.
Plan for conflicts and conflict resolution.

The Lightweight REAIMS Top Ten assessment was done three times during the project – in the beginning of it and in the end of phases 1 and 2 – and it forms the basis for identifying changes in the RE practices. A more detailed infrastructure list (Table 28) was also developed to be able to assess the availability of different elements in the companies.

Both the CMM and SPICE assessments were used to describe the software engineering maturity in the companies but they were implemented very differently. In the beginning, the software engineering practices in the companies were discussed and the CMM levels were estimated informally on the basis of these discussions. Thus the reliability of this estimate is consistent

Table 28. RE infrastructure and their elements

Category	Elements
Templates	Requirements description, Requirements document, Change requests
Processes	Requirements development, Change management, Review
Techniques	Requirements development, Change management
Checklists	Requirements engineering, Change management
Training	Requirements engineering, Software engineering, Application domain
Methods	Requirements engineering, Software engineering
Tools	Requirements management, Change management

with the other assessments that were based on the discussions. The SPICE assessment, on the other hand, was done by a trained assessor. The assessment was conducted to measure five processes in levels 1 and 2 to find possible improvement areas.

8.1.3. Case Study Overview

The first general presentation of RE was given in a tSoft seminar already a year before the evaluation project started. When the evaluation phase approached, three other presentations of the BaRE method were given at two-month intervals before negotiating about testing BaRE in practice. The first quick presentation about the work was given four months before the company specific negotiations; this event was attended by 25 participants from ten companies. Two months before the negotiations, a workshop on the method was arranged, and ten companies expressed their interest in the method by participating in the event. The last presentation was given just before the negotiations to the five companies that were interested in testing BaRE in practice. Consequently, also the initial stage interviews were held with these five companies. However, one company in this group cancelled its participation due to a management decision of not to invest in software process improvement efforts even though the software people saw it important. Another company in this group did not manage to get projects running that suited the method and thus the adoption in this company did not progress during the present project. Finally, three case studies were run between the end of June 2002 and the end of May 2003; to have a common checkpoint it was decided to divide this period into two phases from which Phase 1 (P1) lasted from June to December 2002 and Phase 2 (P2) from January to May 2003. The time before the BaRE evaluation project is referred to as Pre-BaRE (P0) time in the rest of this chapter. The three companies participating in the BaRE method evaluation project are referred to as companies A, B, and C in this chapter.

8.2. Case Studies

In this section, the companies participating in the case studies are first described, then the improvement actions they did are outlined, and the changes in RE practices are reported. The achievement of the original company goals for the project are discussed next, and this section is closed with data on the costs incurred during the project.

8.2.1. Company Backgrounds

Company A. Company A was the IT department of a large metal industry corporation established over a century ago. The IT department was established in the early 1970’s when the first computers were bought in the company. During the present study, the department was responsible both for daily IT operations and software development in the company; seven of the 14 employees in the department were developing software in-house. In addition to their own development, the department also purchased software from external vendors both for internal use and to supplement other products the company produced. To keep this task manageable the department focused on administrative and business applications; the embedded software used in actual products was left for the product development department. The software development normally involved numerous other people from the business and product development departments representing customers and users of the software.

The two key participants in the present project were an IT manager (A1) and a system analyst (A2). The IT manager was responsible for daily operations, software development, and data security, but participated also in some development projects. The system analyst got involved in this project since she was chosen to run the first project with BaRE. The IT manager had 20 years’ experience and a Master’s degree, while the system analyst had four years’ experience and a Bachelor’s degree. In competence evaluation they both selected the Supervised Practitioner level for six areas and the Practitioner level for ten areas (Table 29).

Table 29. Key people in the BaRE evaluation project in Company A

Person	Title	Education	Experience [years]	Competence level				
				N	T	S	P	E
A1	IT Manager	M	20	0	0	6	10	0
A2	System Analyst	B	4	0	0	6	10	0

The main problem in the software development was that everything was a bit unclear. Prime examples of the situation included problems of locating documents after project completion and a lack of requirements test cases. Problems with software project delivery schedules had also resulted in harsh criticism from the steering groups. To improve the situation, an analysis had been conducted and two specific problem areas had been identified: requirements engineering and project management. When seeking for help in these areas, the IT manager contacted the local university and was informed about the BaRE project. The situation in the beginning of the BaRE evaluation project was characterized in short by the IT manager as follows: “As of now we know that something must be done.”

Company B. Company B was a five-year-old company focusing on a single system administration tool. The “single product” refers here to a single source code, since over half of the products sold were actually configured to customer preferences with a product configurator. Lack of customer specific versions was also reflected in the ratio of software developers and other personnel, since only three of the twelve employees were involved in software development.

In Phase 1 there were two key participants: a software manager (B1) and a software engineer (B2). The software engineer had been hired in the company just a month before the beginning of the present project and she continued to study at the local university towards her Master’s degree parallel to working. Her job was to support the software manager in software development, as he was doing both the management and coding parts of the development; the third person in the software development staff was a test specialist focusing solely on testing. The software manager had a Bachelor’s degree and 13 years of experience, while the software engineer had no actual working experience and she had done the studies required for a Bachelor’s degree (Table 30).

Moving from Phase 1 to 2 happened simultaneously with major changes in the personnel. The software manager moved on in his career to another company and soon after that the software engineer decided that it was time to complete her studies, and she also resigned from the company. A new senior software engineer (B3) was hired to manage the software development, while another person was hired to concentrate on software development. The senior software engineer had seven years of experience and a Bachelor’s degree. Considering the competence levels shown in Table 30 they seem to reflect the participants’ experience consistently with higher competence levels achieved with longer experience. Notice also that the senior software engineer had not done requirements engineering himself in his prior assignments.

Table 30. Key people in the BaRE evaluation project in Company B

Person	Title	Education	Experience [years]	Competence level				
				N	T	S	P	E
B1	SW Manager	B	13	0	0	5	9	2
B2	SW Engineer	B*	0	4	11	1	0	0
B3	Senior SW Engineer	B	7	1	9	6	0	0

The RE-related problems were originally summed up as two basic problems: first, documents were not readily available when needed, and second, there had been communication problems between sales and software development people. Later on it was found out that software development knowledge had been limited to one person and documenting the knowledge was important for two reasons: first, to be able to operate efficiently even if that key person was not available, and second, to qualify the company as a training job location for the local university students in computer science. During Phase 1 it also became evident that immature software development processes caused distraction to the developers to the point that something had to be done about it. The managing director for the company commented their overall practices in RE and software engineering as follows: “Our software development will not continue like this.”

Company C. Company C was a six-year-old software house living on bespoke systems development. Only one of the employees was not involved in software development, and even the managing director tried to do his share of the development; all the seven other people were full-time software developers. The application domain for C was administrative and business applications which were often connected to mobile devices for some additional features. Thus the software type was custom software although there was a desire to move to standardized components and even develop own software products.

In Phase 1 there were two key participants: a software architect (C1) and a project manager (C2). The software architect was among the senior people in the company and among the forerunners in process improvement activities. The project manager joined the project a bit later and started studying the BaRE method to be able to use it in his projects. The software architect had a Master’s degree while the project manager had his Master’s degree still under development; in the beginning of Phase 1 he had completed the studies required for a Bachelor’s degree. Both of them had 14 years of experience in software development. In Phase 2 the software architect focused on design and development tasks and did not do RE. However, another project manager (C3) did a project parallel to C2 and thus two projects were run simultaneously also in Phase 2. The project manager C3 had 12 years of experience and a Master’s degree. These key people and their competence levels are summarized in Table 31; again the competence levels appear to reflect the experiences of the people fairly well.

Table 31. Key people in the BaRE evaluation project in Company C

Person	Title	Education	Experience [years]	Competence level				
				N	T	S	P	E
C1	Software architect	M	14	0	0	0	5	11
C2	Project manager	B*	14	0	1	4	6	5
C3	Project manager	M	12	0	1	2	7	6

Discussion on RE-related problems led to more general process problems where three issues were identified. These were the lack of time to document processes, limited sharing of software development and process knowledge, and different working practices between different people. The processes themselves were considered to be clear due to the long experience people had, but a lack of documentation inhibited their wider use and establishment of standard practices in the company. As a minor detail it was mentioned that qualifying as a training job location for university students provided an additional motivation for the documentation effort.

8.2.2. Improvement Actions

Company A focused on implementing the BaRE method in Phase 1 and ran a pilot project in Phase 2. The pilot project focused primarily on getting a touch of the new practices, so Case A describes the implementation of the BaRE method in full scale in a large company. Company B, on the other hand, needed immediate support in their daily software development, and consequently, they took a phased approach to method adoption. The first priority was to solve change management problems, which were addressed in Phase 1. In Phase 2 the focus moved to requirements development, because the planning of a new major software version was initiated then. Thus Case B describes extremely agile software process improvement efforts in a small company living on a single COTS software product. Company C planned to use the BaRE method in full in a single project, but a failure to find a suitable project led to use of only the BaRE RD template. A Finnish translation of the template was done and it was used in two projects. With positive experiences from these projects the template was revised and used in two other projects. These four projects and the RD template adoption process provided many insights about the working of experienced software professionals in a small company focusing on bespoke system development and are documented as Case C. Table 32 summarizes the BaRE method elements and which of them were used by different companies in their RE

Table 32. The BaRE method elements and which of them were used by different cases

The BaRE Method Elements	Case A	Case B	Case C
RD Template	x	x	x
Requirements Development Practices	x	x	-
RM Practices	x	x	-
Tool Support for RM	x	x	-
Training	x	x	x

improvement efforts; in short, Companies A and B used all the suggested elements while Company C used only the document template and training elements of the method.

The projects followed in the study fit the specified domain characteristics fairly well. Only one company did not do administrative and business applications but a system administration tool. The development *teams* included from one to four people and in only one project more than one person was involved in RE. The *duration* of the RE phases varied from one week (W) to five months (MO) but the majority of them were done in four to six weeks; the software development and integration phases together took from about four to eight months. The *effort* of the RE phases ranged from three to five person weeks (PW) with two exceptions – the biggest project comprised of eight person months’ (PM) effort and the smallest one of six person weeks’ effort. The combined effort for development and integration, on the other hand, took from 4.5 months 13 months with the exception of the smallest project that was completed with eight person weeks’ effort. In the end of the present project, two projects had released their software (released or *Rel*), one was in the testing phase (implemented or *Impl*), two in the implementation phase (designed or *Desn*), and one in the design phase (specified or *Spec*). Table 33 summarizes these key characteristics; the project names are formed of the case letter

Table 33. Key characteristics of the followed projects

Project characteristics		Projects					
		A-P1	B-P1	C-P1	C-P2	C-P3	C-P4
Team size [people]	RE	4	1	1	1	1	1
	Development	4	3 (e)	1	2	2-3 (e)	2-3 (e)
	Integration	3	-	1	-	-	-
Duration	RE	5 MO	3.5 MO	1 W	4 W	6 W	5 W
	Development	4 MO	5 MO (e)	5 W	3.5 MO	4 MO (e)	4 MO (e)
	Integration	4 MO (e)	-	6 MO	-	-	-
Effort	RE	8 PM	4 PW	1 PW	5 PW	3 PW	5 PW
	Development	10 PM	11 PM (e)	5 PW	5 PM	4.5 PM (e)	4.5 PM (e)
	Integration	3 PM	-	3 PW	-	-	-
Application Domain		ABA	System Admin	ABA /md	ABA	ABA	ABA
Project Status		Impl	Spec	Rel	Rel	Desn	Desn

followed by “-P” and a running number. The estimated values are indicated with the suffix “(e)” in the table and the “/md” suffix for the application domain refers to additional features implemented for mobile devices.

From the conducted projects, the earliest one was the Company A project which was started in the beginning of September, 2002, and the Company C projects P1 and P2 were started in the middle of September. The later two projects in Company C and the Company B project were all started in January 2003. Looking at these projects, the A-P1 project was quite different from a normal software development project (Table 33). The project was initiated to move a complex database system from old hardware to new and to change the database version, tools, etc. at the same time; an RD was needed to document all the issues that had to be taken care of in the project. All the other projects were normal software development efforts. Company B had released smaller revisions of their software starting from a week’s period to multiple months but major releases were released only every other year or so; the project B-P1 was the first project where a written requirements document was expected to be produced with a major revision of the software. Company C used to have multiple bespoke development projects simultaneously and consequently projects C-P1 and C-P2 were run at the same time in Phase 1 while C-P3 and C-P4 were run parallel in Phase 2. The project C-P1 was so small that it basically followed the waterfall model while in C-P2 specification and development were done mostly parallel. The most interesting feature in C-P3 was that a requirements document had been developed for the desired system as a separate specification project, and thus there was an existing RD to start the project with. However, the customer had found this RD incomprehensible and Company C was contracted to do it again before the actual development could be started. The customer commented on the new RD by saying that they actually understood it with a single reading; the project manager C2, on the other hand, reported that the RD given to him in the beginning of the project created more confusion than helped him. The project C-P4 was implemented in a fairly straightforward manner and the project manager C2 reflected later on this by stating that they would have benefited from early prototyping in the project.

Table 34. Training and information sources utilized by different people (A1, A2, B1, etc.) during different phases

Case Phase Person Information source	A									B				C				
	P0			P1			P2			P1		P2		P1			P2	
	A1	Others		A1	A2	Others	A1	A2	Others	B1	B2	Others	B3	C1	C2	Others	C2	C3
BaRE Guide				x	x					x			x	x			x	x
BaRE Templates				x	x				x	x		x		x	x		x	x
Practical RE in Short				x	x				x			x						x
RE Presentations				x	x	x			x	x	x			x		x		
RE Courses				x	x	x												
Specialist Support	x	x		x	x	x							x	x	x			x
Software Eng. Training	x	x		x	x	x	x	x	x	x	x			x				
Related Literature	x			x						x								

The approach taken in method adoption varied a lot between the companies. Company A chose to implement the BaRE method in full in Phase 1 and ran an extensive training program to support this effort. Companies B and C did not use external support for their effort but focused on getting the practices in daily use in the companies. Table 34 summarizes the information sources different people and companies utilized during the different phases of the project; P0 refers to the time before the BaRE evaluation project, P1 to its Phase 1, and P2 to its Phase 2.

8.2.3. Changes in the RE Practices

The Lightweight REAIMS Top Ten assessment was conducted in the meetings in the beginning of, in the middle of, and after the project (Table 35). Table 35 highlights the difference in RE practices between the phases within each company and, on the other hand, between the companies. A clear change in the practices is shown in the phase the BaRE method was adopted – for Company A in Phase 1 and for Company B in Phase 2. The change was most evident in the infrastructure part of the guidelines (cf. p. 118, the first five guidelines) since after adopting the method companies started reporting having infrastructure available for the named tasks. For example, after adopting the method, companies had a standard document structure defined, the BaRE RD template, which included a unique identifier for each requirement in the standard requirements description templates. In the working practices (the last five guidelines) the change was not so evident since actual implementation of these guidelines depends on the people doing the work and, therefore, it only makes sense to report these practices at most as applied at the discretion of the project manager (Table 8, p. 48). In Company B, two guidelines seemed unnecessary to apply in any way even after the BaRE method adoption. Company C refused to make statements about adopting the method and they consistently reported only the actual changes in their practices. Thus, of the Lightweight REAIMS Top Ten guidelines only the ones concerned with the RD template changed during the project in Company C.

Overall, the Lightweight REAIMS Top Ten assessment provided a quick estimate of the state-of-the-practice in the companies that reflects the actual situation fairly well in these three cases. In Company C the total point gain did not change much (from 7 to 11) during the project, which is consistent with the fact that they had a requirements document template to start with and they focused only on developing a new and better one. However, the changes during the project indicate a move to a more systematic way of working. Companies A and B, on the other hand, had very little to start with and adopted the BaRE method as the method to follow in RE. Since both the companies did introduce many of the suggested guidelines, starting with the infrastructure guidelines, their total point gains experienced tangible increases. It is clear, though, that such a quick assessment is not very comprehensive or reliable, and therefore, a number of other more detailed investigations into the actual practices were performed as part of the present study.

Since the literature based evaluation brought up the difference between infrastructure and working practices as described earlier (p. 118), a more comprehensive list of basic infrastructure was developed and used to supplement the conducted Lightweight REAIMS Top Ten assessments. This list does not focus on RE only, but addresses other areas in software engineering that are inherently intertwined with RE – e.g. change management, reviews, and software engineering in general. The infrastructure list is shown in Table 36. Companies A and B addressed topics that are directly covered by the BaRE method documentation but not much more: neither one of them adopted software engineering methods or acquired training material

Table 35. Lightweight REAIMS Top Ten assessment point gains in different phases. The symbols mean the following: ‘-’ Never applied, ‘O’ Applied at the discretion of the project manager, ‘▶’ Normal use, and ‘●’ Standard use in company

Case Phase	A			B			C		
	P0	P1	P2	P0	P1	P2	P0	P1	P2
Guidelines									
Define practices for RM	○	●	●	-	●	●	○	○	○
Define a standard document structure	○	●	●	-	-	●	○	○	▶
Define unique identifier for each requirement	-	●	●	-	-	●	○	○	▶
Define standard templates for requirements description	-	●	●	-	-	●	○	○	▶
Define validation checklists	-	●	●	-	-	●	-	-	-
Organize informal requirements reviews	▶	▶	▶	-	-	○	▶	▶	▶
Use language simply, consistently and concisely	-	○	○	-	-	○	○	○	○
Make the document easy to change	-	○	○	-	-	○	-	-	○
Use checklists for requirements analysis	-	○	○	-	-	-	-	-	-
Plan for conflicts and conflict resolution	-	○	○	-	-	-	-	-	-
Total Point Gain	4	21	21	0	3	18	7	7	11

in the application domain during the project. Company A arranged some project management training in addition to RE training during the project. Consequently, they also got training material in software engineering from the course; no Company B representative participated in such training. In Company C changes were limited to requirements document and description templates and the adoption of the BaRE method as common training material. Notice that in Table 36, for example, Company C is reported not to have change request templates and requirements development processes even though they are included in the BaRE method documentation; this is caused by an unawareness or lack of interest in such artifacts.

Considering company level working practices, no such clear changes can be claimed even though important improvements were identified. In Company A, the system analyst stated after Phase 1 that they were quite familiar with the techniques suggested in the BaRE Guide. However, after Phase 2 she elaborated the requirements development work from a wider perspective:

My approach to requirements development has changed to a more structured one. How do you actually go about writing requirements? Earlier it was just starting to write the requirements document, but now I really think how things should be and iterate through the requirements to complete the document.

Table 36. Infrastructure in different phases. The symbols mean the following: ‘-’ Not available, ‘○’ Discrepant artifacts exist, ‘◐’ Common artifact available in company, and ‘●’ Standard artifact available in company

Element \ Case Phase	A			B			C		
	P0	P1	P2	P0	P1	P2	P0	P1	P2
Templates									
Requirements description	○	●	●	-	-	●	○	○	◐
Requirements document	○	●	●	-	-	●	○	○	◐
Change requests	○	●	●	-	●	●	-	-	-
Processes									
Requirements development	-	●	●	-	-	●	-	-	-
Change management	○	●	●	-	●	●	○	○	○
Review	○	●	●	-	-	◐	○	○	○
Techniques									
Requirements development	○	●	●	-	-	●	○	○	○
Change management	○	●	●	-	●	●	○	○	○
Checklists									
Requirements engineering	-	●	●	-	-	●	-	-	-
Change management	-	●	●	-	●	●	-	-	-
Training material									
Requirements engineering	-	●	●	-	●	●	-	◐	◐
Software engineering	-	-	○	-	-	-	-	-	-
Application domain	-	-	-	-	-	-	-	-	-
Methods									
Requirements engineering	-	●	●	-	-	●	-	-	-
Software engineering	-	-	-	-	-	-	-	-	-
Tools									
Requirements management	-	●	●	-	●	●	-	-	-
Change management	●	●	●	-	●	●	-	-	-

Company B used the requirements development process only in one project but this was the first software development project where a written requirements document was done. Company B also established new change management practices and processes within a couple of months from the beginning of the evaluation project; they had tried to do this before without success.

Company C did not indicate any big changes in their practices but having only one requirements document template in the company had made it easier to communicate about requirements. The project manager C3 elaborated this topic and the changes in the company more generally as follows:

At least the co-operation has clearly gotten tighter. Namely, now we are thinking together whether things should be done this way or the other while earlier everyone was working alone with the document. Further, everyone was using the template he was accustomed to from previous projects and companies but now we have the advantage that you can ask specific questions and the others understand easier what you mean. ... Of course we had communication also before with the previous templates but not so much and not at the same time as now. Usually the discussions took place after the document was already completed. This made making changes hard while now the discussions take place when the document is still under construction.

Further, a study of the Company C requirements documents in Phase 2 revealed that they both were supported by three appendices. In these projects, two common appendices were a glossary and a use case description document; the actual use cases defined the create, read, update and delete (CRUD) operations for each use case as a novelty.

Table 37 summarizes the properties of the RDs produced in companies B and C during the requirements development phase. Since the Company A project focused on a change of a database implementation, the project differed so much from a normal software development project that a further study of the developed RD is not justified here. Company C translated the RD template to Finnish to be able to use it with Finnish customers and, consequently, the author of this thesis developed a translation of the BaRE RD template to have a similar baseline for both Finnish and English language documents. The *General Properties* in Table 37 refer to the

Table 37. Properties of the produced requirements documents. The counts in parenthesis indicate respective counts in the BaRE RD template. The four last columns refer to the projects that produced the summarized RDs; the first letter indicates the company (B and C) and the suffix thereafter the project number (P1 through P4)

Requirements Documents		B-P1	C-P1	C-P2	C-P3	C-P4
<i>Page count contents/total</i>		12/17	10/14	16/20	23/28	14/19
<i>Appendix count (5)</i>		3	0	0	3	3
<i>Appendix page count, contents/total</i>		58/71	0	0	21/21	34/41
<i>General</i>	Identical	3	0	0	1	3
<i>Properties (9)</i>	Similar	4	0	1	3	1
	Partly similar	2	1	0	1	2
	Different	0	5	5	2	1
	Omitted	0	3	3	2	2
<i>Document</i>	Identical	48	19	12	15	14
<i>Headings (51)</i>	Similar	0	15	24	22	21
	Different	0	13	7	2	2
	Use case name	0	0	6	4	0
	Omitted	3	4	8	12	14
	Count	48	47	49	43	37
<i>Contents for</i>	Group header (8)	8	6	9	8	9
<i>Headings</i>	Exists (43)	34	32	21	35	28
	Missing	6	9	19	0	0

existence and contents of a cover page, document information table, change history, table of contents, document template (headers and footers), appendices, requirements tables, other tables (Stakeholders, Product Position Statement, Document Overview, Users, Important Domain Properties, Database, Documentation Requirements, and Likely Changes), and diagrams. The *Document Headings* refers to the headings of the RD template and the *Contents for Headings* refers to the actual text or requirements under each heading; the *Group Header* and *Missing* indicate the number of headings without any data. None of the produced documents had *General Properties* or *Document Headings* that were not suggested in the BaRE template or were not comparable with some other item in it.

Table 37 reports also the differences between the produced requirements documents and the BaRE requirements document template. The counts in parenthesis in the table indicate the respective item counts in the BaRE template; *Identical* means that no differences could be identified between the requirements document under study and the BaRE RD template while *Similar* refers to practically comparable meanings. *Partly similar* refers to an identifiable difference between the items and *Different* refers to a clear semantic difference between them. *Omitted* is used to record the count of items removed from the BaRE template and the *Use case name* reports the count of use case names included in the table of contents since one project manager was accustomed to do so.

Looking at Table 37, it is striking that the requirements documents are fairly short and appendices tend to include a wealth of details. The three projects reporting use of appendices had all produced a Use Case Description document and two of them had also prepared a glossary; the third project reported a need and intent to make a glossary also but had not yet managed to do so. The third appendix was different in all the projects – one project had a Detailed Requirements appendix, another had a Report Examples appendix, and the third project had put a customer supplied interface specification to an adjacent system as an appendix to the RD. Another general observation from Table 37 is the fact that the Company C documents produced in Phase 2 (C-P3 and C-P4) resemble the BaRE RD template much more than the earlier projects produced in Phase 1 (C-P1 and C-P2). The fact that the C-P3 and C-P4 documents appear fairly similar to each other follows from the fact that the project managers for these projects made a conscious effort to develop a common RD template for the company during these projects; some of the differences, on the other hand, followed from the fact that the C-P3 project used Microsoft Word for the text processing while C-P4 used LaTeX for this purpose. Thus, even though a serious effort was made to establish a common RD template in the company, some infrastructure issues still remained to be solved.

The requirements documents summarized in Table 37 followed the BaRE RD template development ideas by and large. Company B adopted the English RD template and made two kinds of changes to it. First, some company specific adaptations to administrative data and its representation were made, and second, three sections were dropped from the contents: document structure, users, and business process description.

Company C, on the other hand, made changes to all the three areas reported in Table 37 – *General Properties*, *Document Headings* and *Contents for Headings*. The general changes included dropping the use of tables for requirements and other information since plain text was seen as a better way to manage large amounts of data (i.e., *Omitted* in Table 37). However, the textual information had a standard structure and in most cases the text was preceded with a tag like “Description” or “Format” to identify the purpose of the data. The *Different* items included

a change to a company specific look for the cover page and a different kind of diagram use in the document while the *Partly Similar* items covered appendices and further deviations in the diagram use. The project C-P4 left a total of 14 of the BaRE template headings away from its own document; C-P3 left also 12 headings away but it had both one similar and one identical heading more than the project C-P4. The headings left away by both the projects included the ones describing product, document structure, logical user interface characteristics, business process description, performance, reliability, scalability, and documentation requirements. C-P3 had come up with the same translation for the Miscellaneous Requirements heading as in the BaRE template and used a similar term for the Design Constraints headings while C-P4 omitted them both totally. The different approaches for the Miscellaneous Requirements heading could well be connected with the decision to use the phrase “additional requirements” as the translation for Non-Functional Requirements, as this phrase is quite close to the phrase “miscellaneous requirements.” The biggest differences in the headings, however, occurred with the translations of the headings Requirements Allocation and Availability as the Company C translations for these terms were phrases “implementation baseline” and “level of use,” respectively. As the last point, the differences in the group header deserve two comments. First, the differences in the group header counts were caused by some descriptive text that was added in the project C-P3 document with little practical relevance for the document itself. Second, neither one of the Company C projects had headings without contents which made the documents look like completed ones.

As the last point in this section, the results of the SPICE process assessment are summarized. This assessment was conducted independently from the present study during its final month. The purpose of the software requirements analysis process is “to establish the requirements of the software components of the system” (ISO/IEC TR 15504-5 1998, p. 20) and the assessment results were the following:

Company A: “Process assessment was conducted on levels 1 and 2 and level 2 was achieved. Level 1 assessment covered performance of 8 base practices, and as most of these practices were fully achieved, the overall rating was that the level 1 performance attribute was fully achieved. Level 2 assessment covered 8 management practices both in performance and work product management attributes, and the level 2 was fully achieved” (Saastamoinen and Tukiainen 2003a, p. 18).

Company B: “Process assessment was conducted on levels 1 and 2 and level 2 was achieved. The company had clearly defined a requirements management process and the level 1 performance attribute was fully achieved. Also level 2 management practices were achieved largely both in performance and work product management attributes. The company used the BaRE method to elicit and specify requirements and managed them with the *sfrm* software. These tools made it possible to achieve an established and predictable way of working and maintain requirements documents” (Saastamoinen and Tukiainen 2003b, p. 16).

Company C: “A software requirements management process assessment was performed and level 1 was achieved. The assessment covered 8 base practices and most practices were fully or largely achieved. The most significant omission on level 1 base practices was lack of measurable quality requirements and lack of procedures to handle requirements change management and traceability; thus

these two practices were achieved only partly. Level 2 assessment covered 8 management practices both in performance and work product management attributes. The performance attributes were achieved in general largely, while the management practices appeared to be less well addressed and the management attributes were achieved only partly” (Saastamoinen and Tukiainen 2003c, p. 18).

As a summary of the SPICE assessment, the overall requirements management process capabilities in companies A and B were rated at level 2 while Company C was rated at level 1. This result is comparable with the outcome of the Lightweight REAIMS Top Ten evaluation (p. 138) where companies A and B achieved 21 and 18 points, respectively, and Company C got only 11 points.

8.2.4. Achievements

None of the companies participating in the project were really after point gains in any assessment but wanted to improve their software and RE practices. The achievements in this respect are summarized in Table 38, which shows the goals and achievements for each company and phase. As a whole, the companies achieved their goals fairly well even though the improvement efforts were not completed during the present project. However, the achievements were encouraging and all the companies planned to continue the use of the adopted practices and extend their use wherever possible. For example, the senior software engineer in Company B stated that they do not have many possibilities to extend the use of the BaRE method but they plan to deepen the use of it. Considering Table 38, it should be noted that the goals for Phase 2 were defined only in the end of Phase 1; this situation is shown in Company C goals that were scaled down after the experiences from Phase 1. For the definitions of the used terms (Defined, Available, Trained, and Adopted) see Section 2.4 (p. 32).

Table 38. Goals and main achievements in different cases and phases

Case	Goals	Main Achievements
A/P1	<p>Document templates are defined and available. The standard location for documents is defined. A requirements development process and techniques are defined. A change management process and procedures are defined. A requirements management tool is adopted. People know how to use all new RE artifacts.</p>	<p>Requirements document, requirements description, and change request templates adopted. Standard location for requirements documents defined as the RM tool. Requirements development practices defined: process, techniques, and tasks. Change management practices defined: process, techniques, and tasks. A requirements management tool adopted. People trained in RE.</p>
A/P2	<p>Test the new practices with a pilot project.</p>	<p>A project swapped to the new practices in the end of requirements development phase and requirements completed with new practices; in the end of development phase the project appeared successful.</p>
B/P1	<p>Document existing requirements in a common format and place them in one data store. Define change management practices.</p>	<p>Existing requirements documented in a technical specification. Change management practices defined and adopted. A requirements management tool adopted to support the change management. Control over changes established.</p>
B/P2	<p>Adopt a requirements document template. Adopt requirements development practices.</p>	<p>Requirements document template adapted and adopted. Requirements development practices adopted.</p>
C/P1	<p>Define requirements document template(s). Define requirements development process.</p>	<p>A Finnish RD template defined based on the BaRE RD template and tested independently in two separate projects.</p>
C/P2	<p>Spread knowledge about the new RD template and get it in a wider scale use in the company.</p>	<p>An RD template defined as a co-operative effort in the company. Defined RD template used in two different projects.</p>

8.2.5. Costs

To get an idea of how much the RE improvement efforts cost, data on the time used was collected from the participating companies. In the beginning of the project the companies were asked to record the time they used in RE related activities during this project, and this information was collected systematically in every meeting. Only one person (A1) recorded the time use in a weekly time sheet and gave it for further analysis; all other people provided an estimate of their time use without looking at any archival records. However, many of the activities (i.e., *tSoft*, *RE Info*, and *Training*) were scheduled by organizers, and the time used for *Meetings* was recorded by the researcher. Table 39 summarizes the time use in the three companies (Cases A, B, and C) and reports also how different persons (A1, A2, etc.) used time for the activities of interest during this project. The *tSoft* column includes the time used for tSoft-project activities, i.e. seminars on software engineering and the SPICE assessment conducted in the companies. The *RE Info* column includes the time used for RE specific presentations and a workshop while the *Meetings* column shows the time used for the BaRE evaluation project meetings; these two activities also summarize the time spent with the author of this thesis. The *Training* column summarizes time use for four different courses: a two-day course (15 hours) on software process improvement for persons A2, one person in the group A-Others, B1, and C1; a four day course (30 hours) on project management for A1, A2, and 18 people in the A-Others group; a two day course (15 hours) on the DOORS tool for A1, A2, and five people in the A-Others group; and a requirements writing course (4 hours) for A1, A2, and 19 people in the A-Others group. The *Study* column shows the time used for personal studies on RE and the BaRE method; the IT manager A1 did study literature with a broader scope including books on project and risk management as well as process improvement in general. The *RE Dev* column reports the time that was used in RE related development activities and the *Total* column summarizes the time use for the different people and their companies.

Table 39. Time use in hours for RE and process improvement related activities

Case	Person	tSoft	RE Info	Meetings	Training	Study	RE Dev	Total
A	A1	10	3	16	49	140	162	380
	A2	4	3	15	64	15	74	175
	A-Others	7	13	16	706	0	127	869
	Totals [h]	21	19	47	819	155	363	1424
B	B1	1	1	10	15	3	0	30
	B2	4	2	5	0	21	32	64
	B3	7	0	9	0	21	14	51
	B-Others	14	9	7	0	0	0	30
	Totals [h]	26	12	31	15	45	46	175
C	C1	8	6	7	15	3	6	45
	C2	7	0	7	0	3	23	40
	C3	0	0	3	0	2	40	45
	C-Others	11	1	2	0	0	0	14
	Totals [h]	26	7	19	15	8	69	144

Looking at Table 39, Companies B and C applied fairly similar effort in the improvement efforts – about one person month – and in the same vein neither one of them spent any direct money on the method adoption and improvement efforts. The BaRE evaluation project itself was free for the participating companies but since it was a part of the European Union supported tSoft project, the companies had to pay a small self-financing fee for the tSoft membership. Company A, on the other hand, approached the improvement effort from quite a different angle and spent an effort equal to about nine person months on it. It must be noticed, however, that this time also includes about four person months of training (630 hours) in software process improvement and project management. Company A also spent money directly on a commercial RM tool, RE training, and consultation. The calculatory costs for the time used in the improvement effort were about equal to the tool costs or training and consultation costs. Consequently, the total costs for the improvement effort were formed from these three fairly equal costs for Company A. Due to the large investments in the effort, return on the investment calculations were made; these calculations focused only on the tool purchase and suggested a two year return time for the investment.

8.3. Contexts for Change

In this section, evidence is sought to identify the factors that contributed to the changes in RE practices in the present study. This is done by first introducing a framework that is used in the evaluation to separate affecting factors. After that the evaluation is reported from three viewpoints identified in the framework: environmental context, organizational context, and technological innovation. This section is closed with a summary on the factors affecting the study outcomes.

8.3.1. Evaluation Framework

In Chapter 5, thirteen characteristics were identified for the solution that provided a basis for solution development but for its empirical evaluation they provide only a limited and subjective basis. To achieve a broader and more objective basis for the solution evaluation, a framework suggested by Tornatzky and Fleischer (1990) was adopted. This framework has been developed to identify reasons that lead a potential user to become a real user of a technological innovation by studying technological innovation decision making from three viewpoints: the environmental, organizational, and technological contexts (Figure 27).

Tornatzky and Fleischer (1990, p. 153) define the environmental context as industry, competitors, access to resources supplied by others, and dealings with government. This follows from an observation that industry members, knowledge producers, regulators, customers, and suppliers can influence the degree to which a firm (a company) sees a need for, seeks out, and brings in new technology. The environment can both provide opportunities for innovations through innovation-related information, financial and human resources but also constrain possibilities through government policies, regulations, capital availability, and restrictions on information flow. As the organizational context, Tornatzky and Fleischer consider company size; the centralization, formalization, and complexity of its managerial structure; the quality of its human resources; and the amount of slack resources available internally. The organizational context also includes the informal linkages between employees and the transactions carried through them which mean internal decision making and internal communication. Finally, the technological context is summarized as the internal and external technologies relevant to the

company including current practices and equipment internal to the company as well as the pool of available technologies external to it. As the last point in Figure 27, it is noted that the double headed arrows in the figure acknowledge the boundary spanning mechanism evident between the different contexts.

The Tornatzky and Fleischer framework was adapted quite a bit for the needs of the present study. First of all, the environmental and organizational contexts were modified to use the names given in the text of the book (Tornatzky and Fleischer 1990, p. 152) and the technological context was given a new name. Namely, Tornatzky and Fleischer (1990) wanted to emphasize the role of technological context by separating it from the environmental context: “we consider it [the pool of available technologies external to the company] separately from the rest of the environment in order to focus attention on how features of the technologies themselves can influence both the adoption process and implementation” (p. 153). In the present study this focus was better brought up by naming the technological context as *Technological Innovation* and focusing on its characteristics at a more detailed level. Furthermore, in the present study the framework was defined to identify factors affecting the outcomes of the study which could be either adoption of the BaRE method, or some part(s) of it, which resulted in changes in RE practices in the adopting company (Figure 28).

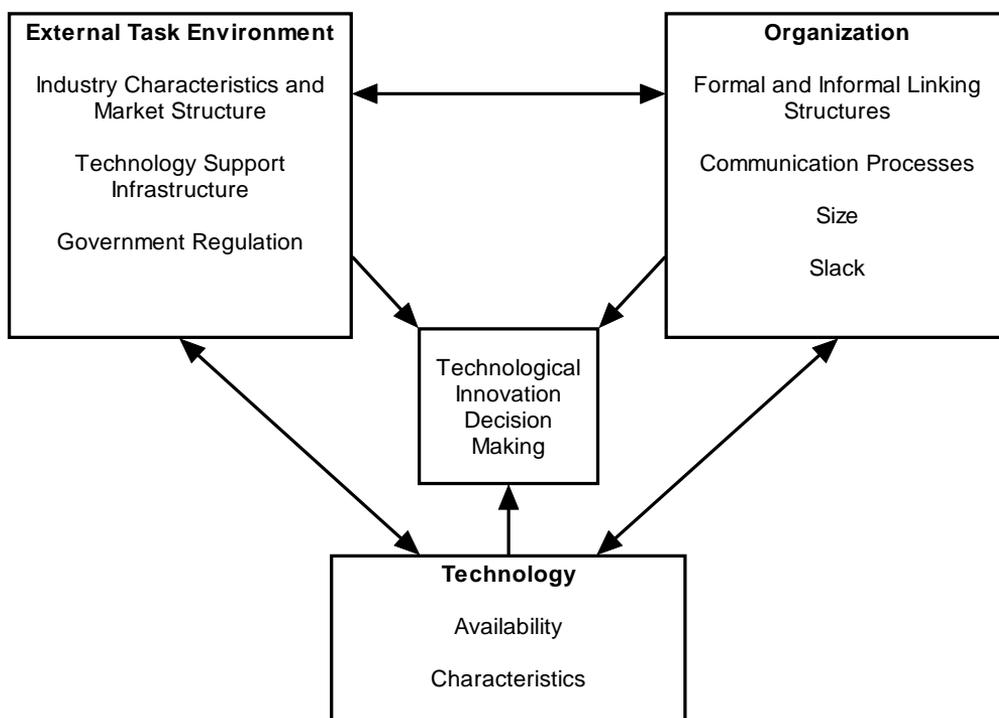


Figure 27. The context of technological innovation (Tornatzky and Fleischer 1990, p. 153)

The smaller adaptations of the framework were numerous (Figure 28). Due to the selected approach, the *current practices* and *current equipment* were excluded from the aspects of interest since they were effectively addressed by the *compatibility* characteristic of the innovation, and the organizational context was extended by adding a new topic *other internal SPI efforts* to it to be able to control the possible effect of related efforts in the organization. In the environmental context the dealings with government and regulation were considered irrelevant in the present study since software process improvement in general has little to do with them. Bearing in mind that the anticipated method was targeted for administrative and business applications there were no regulations on the software development process and, on the other hand, the small scale focus of the approach made any connections to government unjustified. Two other changes due to the small scale focus of the present effort were changing the phrase *knowledge producers* to *external information sources* and instead of *supplier* to talk about *business partners* in general. In the original framework, the organizational context was

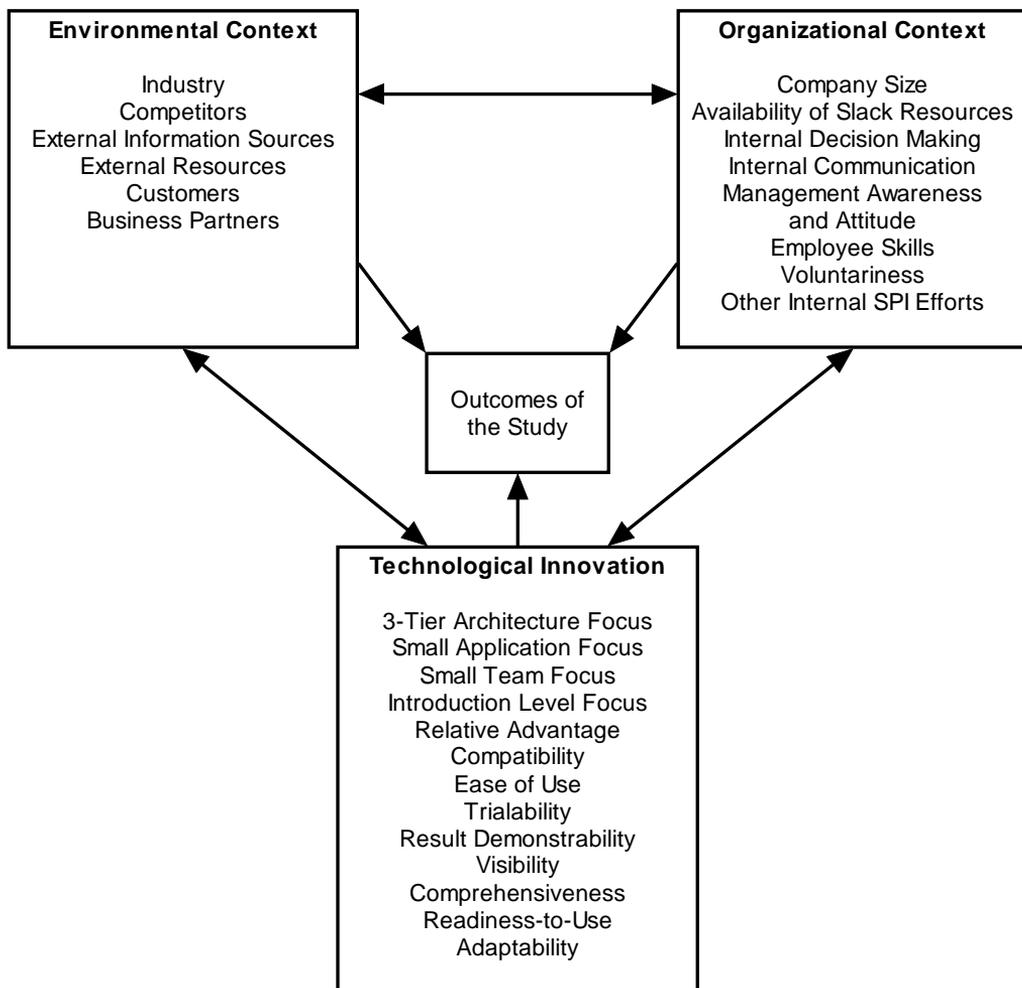


Figure 28. The empirical evaluation framework of the present study, adapted from Tornatzky and Fleischer (1990, p. 153)

claimed to be influenced by *the centralization, formalization, and complexity of its managerial structure* but the present focus on small companies made the fit of this approach questionable. Therefore, it seemed better to focus on *voluntariness* (Table 12, p. 58) supplemented with *management awareness and attitude* to capture the management aspects on the technological innovation decision making. In the same vein, *quality of human resources* seemed too advanced and it was replaced with *employee skills*.

8.3.2. Environmental Context

The environmental context of the present study consists of industry, competitors, external information sources, external resources, customers, and business partners. In the present study the most important of these factors seemed to be the external information sources. Company A also utilized an RM tool consultant and literature to complement the information provided by the author of this thesis. The starting point for the Company A RE development work was a need for a long lasting and reliable database-based requirements management tool due to the nature of the company's basic business. Therefore, the central requirements for the tool vendor were an established position in the market, reliability, and longevity. After the vendor selection, a senior consultant was responsible for the necessary adaptations to the selected RM tool so that the document structure suggested in the BaRE method could be implemented. At the same time suggestions were also made about the process and general handling of the requirements meaning that this outside source was an important factor in the study outcome in Company A. This importance is further supported by the fact that Company A's idea of improving the RE practices was to get an established RM tool in place with a working RE process and get them to work together. The role of the literature in Company A was mostly providing reassurance that the practices and ideas suggested in the BaRE method actually presented viable and established practices in RE. As a whole, Company A brought an RM tool in the company with the help of a consultant and thus this environmental factor is more accurately described as a business partner, or supplier, affect. The process or method, on the other hand, was then introduced in the company with the help of the author of this thesis.

The external information sources used by the two small companies (B and C) were limited to the tSoft program and the author of this thesis through the BaRE evaluation project. Since the BaRE evaluation project presented the RE area in the tSoft project, the RE specific external information sources were limited to the author of this thesis in these two companies. More general information about SPI was gained through the training arranged by the tSoft project and the university software engineering courses attended by the software engineer in Company B. Hiring a new senior software engineer was another potential external information source for Company B, but this person indicated in an interview with the author of this thesis that in his previous assignments he had only been using requirements documents and that he had no experience in developing one.

Since the problem of a researcher participating in the SPI effort was a known limitation of the study from its very beginning, two actions were made to control this effect. First, all the information and support provided by the author of this thesis to the participating companies were based on the BaRE method documentation. Second, the role of the author in the study was questioned from the companies in the end of the study. To summarize the expressed opinions, all the companies stated that an outside participation was important for the success of the effort and the results that would have been achieved without the author of this thesis had not been as

good as they were now. For example, the senior software engineer in Company B commented this topic in the following way:

The effort would probably have succeeded in some respects but not in the present extent, detail, and schedule. The external support was most valuable and necessary especially when the development personnel of the company changed in the middle of the project.

This technical viewpoint was complemented by the system analyst in Company A who commented this topic more from a champion point of view (Humphrey 1989, p. 31). Namely, the reply to a question what would have happened if the author of this thesis had never visited the company was the following:

It could be that the BaRE method documentation would have just circulated on the tables. I have a feeling that your visits somehow made us work on RE, study it, and try to understand it. So it is my feeling that the visits made a difference in comparison with the situation that we would have studied the documentation only by ourselves.

One should also notice that the fact that this evaluation project was an academic piece of work made an important contribution to its overall trustworthiness. In addition to the author of this thesis, the overall project involved two universities, one academic advisor in close co-operation in getting the industrial projects started, and numerous other researchers connected through the tSoft project. The academic nature of the work was commented on explicitly only by the IT manager in Company A as follows:

In my opinion academic nature of a work makes an important contribution to its value. I do look at a document itself to find out its value, but a document or book on some topic certainly does not have the same value as it would have as a licentiate or doctoral thesis. They do have much higher value as such.

The other factors in the environmental context include industry as such. From the three participating companies, two small ones were software companies and the third one, the large one, was from metal industry. Thus, even if software industry is promoting process improvement etc., there is no evidence that the industries as such would have affected this study and no references to industry bodies or alike were made during this project. The impact of business partners, customers, and competition in the present study seemed low in general. In Company B, the effect of external organizations appeared to limit to the need to be able to manage their change requests and requirements for the COTS product in a more efficient way. Company C ran bespoke system projects and customer feedback on produced requirements documents was one important reason for improving them in the first place, and on the other hand, the company engaged in a partnership with another company, which created a need to communicate about system requirements with a partner as well as with customers. Company A had two kinds of business partners in the software engineering area: ones that delivered software to them and others that helped in the SPI efforts. From the point of view of the present study, these partners in the SPI area provided a kind of benchmark since the IT manager reported that the BaRE method provided a better fit for the company needs than the practices used by these partners. The partners providing software development services, on the other hand, were reported in many cases to lack systematic practices which made it necessary for

Company A to establish their own practices to use as a benchmark for their software providers. Finally, the tSoft project did provide access to some other software engineering related resources (mainly information in web and consultancy), but use of these resources or other external resources was not reported by the participating companies. Thus it is justified to claim that other external resources did not play any real role in the present study.

Considering the environmental context of the evaluation project, it seems clear that only external information sources and business partners had an effect on the outcomes of the present study. In Company A the tool vendor helped in many concrete ways to change the practices and all three companies considered the author of this thesis as an important contributor to the changes in the companies. Thus there is evidence to support the claim that the availability of these two people and the organizations they represented had a direct influence on the outcomes of the present study.

8.3.3. Organizational Context

The organizational context of the present study included eight factors: company size, availability of internal slack resources, internal decision making, internal communications, management awareness and attitude, employee skills, voluntariness, and other internal SPI efforts. In this section these factors are described in this order and the section is closed with some observations on other factors affecting the innovation decision making since they are closely related with the organizational context of the study.

Considering the characteristics of the three organizations in the present study, they all had approximately ten people in the software related tasks. The software development teams were generally small and had only one to four people (Table 33, p. 135) but otherwise the companies were quite different: A was an IT department of a large metal industry corporation, B was a software product house, and C focused on bespoke software development. Thus Company A called most of their software people business analysts, C called them software developers, and in Company B most of the people were sales people. Common to all companies, however, was a lack of internal slack resources which was also one of the starting points for the present study (Section 4.2.2 Lack of Resources, p. 81). The internal decision making seemed to be efficient in the small companies where the managing director made decisions quite quickly based on informal suggestions and justifications. The large company seemed to operate more slowly and formally in this respect since decisions were made in different management levels considering different aspects like other corporation companies and return on investment calculations. The company sizes were also reflected on the interest in making investments since the small companies showed clear reluctance to make investments but the large company made a major investment before they could show any concrete benefits from it (Section 8.2.5, p. 145). Finally, the internal communication was implemented in companies A and B as weekly meetings were central issues were discussed. In Company C no such practice was observed or mentioned but informal discussions aside the work and during coffee breaks were brought up. Overall, the organizational characteristics of these three companies were very different and resulted in very different improvement approaches.

The management awareness of the problems in RE and software engineering in general was evident from the beginning in all the companies. In the same vein all the managers expressed their support for the SPI effort even though in practice this support was showed very differently.

In the two small companies the managing directors expressed interest to the study throughout it, but in practice there is no evidence that they actually did anything concrete to advance it or the RE practices in the companies. This outcome is not a surprise since managing directors are responsible for running the companies, and delegating SPI efforts to appropriate senior people makes sense. In the large company, on the other hand, the IT manager had a very active role in the SPI effort communicating about the project with numerous stakeholders and arranging training, external consultancy, tool purchase etc. Overall, the managers can be claimed to have acted as one could expect – the managing directors in companies B and C operated as the project sponsors who are in authority and can recognize the value of the work and sponsor it (Humphrey 1989, p. 31) while the IT manager in Company A operated as a champion who brought “the management’s attention to the subject, obtained the blessing of a sponsor, and established the credibility to get the change program launched” (Humphrey 1989, p. 31). The IT manager seemed to have an additional role as the change agent who led the change planning and implementation.

Both the employee skills and the company RE practices were studied during this evaluation project. The employee skills were surveyed in the beginning of the study (Table 29-Table 31, p. 132-134), and based on this survey the skills varied quite a bit between different people. On the company side the first Lightweight REAIMS Top Ten assessment before the BaRE evaluation projects suggested that the RE practices were initial by and large (Table 35, p. 138). This outcome is further supported by the infrastructure studies (Table 36, p. 139) and the statements the IT manager in Company A and managing director in Company B made about their overall situation in software engineering (Section 8.2.1) which can be summarized as a clear need for improved practices. Company C represents the most advanced organization in the RE practices in the present study, but even though the company people seemed advanced, the common infrastructure – especially document templates and practices – was at the initial level. It should also be noted that in all the companies the provided method documentation was used to increase understanding of RE. In Company A a roadmap to improve the RE practices was developed and the value of RE documentation was realized in the course of this project. Further, Companies A and B reported that the increased knowledge in RE had eased the requirements development and management work in practice. This was expressed clearly by the IT manager in a meeting just before closing the present project:

Now we know what requirements engineering is about. Now we know when we have requirements collected and we also understand why things have been so problematic before. Further, now we know how to make things work. Now is the time to implement the changes.

The voluntariness of the method use was not questioned as such in the present study but even though all the managers expressed a need for improving the existing practices, the present study did not find any indication of the new method use having been mandated. In the two small companies the managers appeared to rely on the people using the method and the operative decisions appeared to have been left totally up to them. In the large company, on the other hand, the voluntariness was discussed explicitly with the author of this thesis with the following comment by the IT manager:

One thing that I have been thinking about is how we could change the way people work without always having to force the change. How could you change the way

people work without always having to use a mandate – that people would have a desire to change their way of working instead?

This discussion took place in the end of the evaluation project when the possibilities for speeding up the new change management procedure adoption were explored, and the management mandate use was brought up as the last resort in getting the people to follow the new procedure. However, this option was not used during the evaluation project and it was evident in all the participating companies that all the people looking at the proposed method did it with the goal to improve the RE practices and the overall quality of the work in the companies.

The other internal SPI efforts refer to the potentially existing other internal process improvement activities. As was explained in Section 8.2.5 (p. 145), all the three companies participated in process improvement training during the evaluation project but few changes – except in RE practices – were identified by the end of the project. Company C started revising their other document templates (project plan, change requests and changes –documents, and contracts) but these changes were based on the experiences of the requirements document template development work rather than vice versa. Company A, on the other hand, developed plans to improve their project management and test practices; since the selected RM tool provided a possibility to document requirements test cases, they were developed as part of the first project. Company A arranged also a project management course (Section 8.2.5, p. 145), but no tangible results could be identified from these efforts by the end of the present evaluation project. Thus no other SPI efforts were identified that had affected the BaRE method evaluation project outcome. Instead, the opposite evidence was encountered as the IT manager in Company A mentioned six months after the project management training that so far it had not resulted in concrete changes in the company practices.

Section 3.2.5 (p. 55) summarized a number of other factors that have been found important in technology adoption efforts. In the BaRE evaluation project these factors were not considered very important a priori but some observations were made about them in the course of the project. In Company C the *subjective norm* was mentioned when it was discussed who gets to do what in the company: when the project manager C3 was told to start using the BaRE method in his project, an informal discussion indicated that the most experienced people in the company got to do the most interesting stuff. Even though this statement was presented as a joke, it bears an indication that there were authorities in the company whose opinions were respected. This observation is closely related to the discussion above on management awareness and attitude which brought up the roles different managers had in the present study. In Company C the software architect clearly complemented this group by being the champion of the project – even though he was not in a management position. In Company A, on the other hand, the *social pressures* from the older staff members were expressed to reduce the enthusiasm of the system analyst towards the new requirements template. At least on one occasion the system analyst reported to have expressed confusion about what to write under some specific heading in the document, and the following laconic statement was put forward: “one day you will learn that it is not worth considering those things so precisely.” Common to all these companies, however, was that all the people were concerned about the *perceived behavioral control* in the method adoption meaning that there was a clear lack of time to learn the method and take it in use on a wider scale. This observation is clearly related to the lack of slack resources discussed above. As all these discussions were conducted long after the initial adoption decision was made, there is no reason to claim that they would have affected the decisions. However, these examples

demonstrate the importance of the organizational factors⁶ as they may affect the continued use of the method in the companies.

Considering the organizational context of the present study there is little evidence to support a claim that some of the studied factors would have played a central role in it. Paying special attention to team sizes and RE practices it became apparent that these companies operated with small teams and the level of RE practices in general was in introduction level. From the other factors only the management awareness and attitude to the improvement effort and, more specifically, the named sponsors and champions for the present effort appeared to have direct influence on the outcomes of the study.

8.3.4. Technological Innovation

In this section the thirteen characteristics of the BaRE method are studied more closely to identify their effect in the present study. Since no definitive order of importance was identified in the course of the study, the characteristics are discussed in the same order as they were introduced in Chapter 5. For each characteristic the same principle topics are discussed: a brief definition, a summary of how the method was expected to address the characteristic, observations from the followed projects to validate the presence of the characteristic and, on the other hand, its effect on the study outcomes. Since the effect is seldom easy to identify and justify, a scale from *direct influence* through *indirect* to *no influence* is used. This section is closed with a summary of the factors that affected the study outcomes.

3-Tier Architecture Focus. 3-tier architecture was defined to refer to administrative and business applications with a user interface, application or functional unit, and database management system (p. 90). The requirements document template has a number of specific sections for different user interface properties: listing all user interfaces by name, their logical characteristics, product functions described as use cases, and user category descriptions. The application subsystem or functional unit is also described from different perspectives: important domain properties, business process, and product functions as use cases. Finally, the database requirements are described by a data model and database requirements in addition to the general performance requirements. For each of these three subsystems, a number of more detailed descriptions are suggested in the interface specification not necessarily understandable by end users. For example, the interface specification is suggested to include general user interface development guidelines, dialog maps, detailed window descriptions, refined data models with details, administrative user descriptions, and a CRUD (create-read-update-delete) matrix for database operations. Thus the selected architecture is fundamentally reflected on the documents defined for the target application domain. In the same vein, techniques for eliciting business and administrative requirements are suggested in the method including document studies, apprenticing, interviews, and brainstorming.

⁶ To be precise, subjective norm and perceived behavioral control are more psychological than organizational factors, but since the former is missing in the selected evaluation framework the organizational factors are used in the present study.

The 3-tier architecture focus proved to match the needs of Companies A and C fairly well. Namely, both of these companies reported to focus on administrative and business applications (Section 8.2.1, p. 132-) and all the four studied Company C projects included requirements in all these three areas (Table 33, p. 135). Since the Company A project focused on upgrading the company database, it was not a representative project in the company and does not provide evidence of a typical application domain in the company. The Company B focus on the system administration tool was quite different from the original BaRE method application domain, but the method proved to serve its purpose well in the present study. The senior software engineer summarized this perception in the following way:

[The BaRE method] has been quite suitable for our application domain. However, once we get a bit further in this project we shall know better if something is missing from it. So far nothing essential has been missing.

This view was supported by the fact that in a review of the developed requirements document no major problems were identified in the document. This review was conducted by five people in addition to the author of the document, took two hours to complete, and identified altogether 23 issues in the document.

From the BaRE method design point of view the 3-tier architecture focus was a reasonable decision since all the participating companies used the elements it covered. However, even though the method seemed to fit these kinds of projects, the way this characteristic actually affected the outcomes of the study was not measured directly and thus there is no way to estimate its effect objectively. It can be speculated, though, that the specific architecture made it possible to, for example, increase the ease of use and readiness-to-use considerably in respect of an approach without any application domain constraints. Thus it can be concluded that even though the 3-tier architecture focus characteristic cannot be claimed important direct influence on the study outcomes, it seems reasonable to claim that it had indirect influence on the outcomes.

Small Application Focus. Small application focus was defined to refer to a project having one requirements engineer, less than half a dozen developers, a duration from six to twelve months, and a budget below 100 kEUR (p. 90). This focus is reflected in the BaRE method by the fact that word processor templates are expected to prove valuable in the development phase. Large scale application projects, on the other hand, cannot be expected to benefit from a simple document template very much since they often rely on distributed architectures, introduce complex and nonnegotiable constraints between different systems etc. These and many other properties natural for and inherent of the large applications are not addressed in any way in the BaRE method making the fit of the method for such applications questionable. However, with small applications the need for these kinds of capabilities should be much lower making the BaRE method suitability to small applications a realistic goal.

All the projects followed were fairly small in size, starting from less than two person months to 21 person months (Table 33, p. 135). The software architect who implemented the smallest project commented on the importance of documenting requirements in a small project as follows:

In my opinion even small projects need written requirements documents. Even if a six person week effort is a small one, in a smaller company it is still relatively significant with respect to the company size.

The project manager C2 supported this view by stating that he could not figure out a project where written requirements document would not be needed. Thus it can be concluded that small projects need support and advice on systematic RE for the same reason large and complex projects need it – it is an important way to reduce risks involved in projects.

Due to fact that the method was used in small application projects it can be claimed that it actually fitted such projects in the present study. Despite this, it seems hard to claim that this characteristic would have played an important direct role in the study outcomes. It can be speculated, though, that the small application focus made it possible to, for example, increase the ease of use and readiness-to-use considerably in respect of an approach without any project size constraints. Thus it can be concluded that even though the small application focus characteristic cannot be claimed important direct influence on the study outcomes, it seems reasonable to claim that it had indirect influence on the outcomes.

Small Team Focus. The small team focus was defined to refer to projects that have less than half a dozen people and one requirements engineer is responsible for and in charge of the development of every requirements document (p. 91). The assumption that a single person documents and manages the requirements for a single project provides the justification for two other ways to simplify the method. First, a word processor document template is suggested to support requirements development which is unlikely to be very reliable or practical in a large team environment. Second, no techniques are suggested, for example, to identify inconsistencies between requirements or conflicts between stakeholders etc.

As it was noted above in the organizational context, all the followed projects were conducted with small teams. However, in Company A multiple people were reported to write requirements even if in every project basically one person operated as the principal requirements engineer. This approach was also reflected in the decision to adopt a commercial requirements management tool. In Companies B and C, on the other hand, in every project only one person was doing requirements engineering and requirements were documented in text based documents in the requirements development phase.

It seems evident that the small team focus as such had little direct influence on the study outcomes. However, the document based approach can be suspected to have had indirect influence through increased compatibility with previous practices, ease of use, and readiness-to-use in the small companies as it made the need for initial investments and changes limited.

Introduction Level Focus. The introduction level focus was defined to refer to basic systematic RE practices (p. 91). This is reflected by the fact that the central elements of the BaRE method are documenting requirements and change requests which should make it easy to improve and refine the descriptions as understanding of them increases.

The summary on the organizational context and skills of the people participating in the present study (p. 152) suggest that the introduction level focus matched the practical situation by and large. For example, all the organizations developed their requirements document templates in

the course of the study and provided training in RE or studied it themselves (Table 32, p. 135). Furthermore, two companies revised and documented their requirements management practices while in the third company it was also noted that there is evident room for improvement in this area.

Despite the benefits the introduction level focus provided for the adopting companies, there is no evidence to support the claim that this focus influenced their decision making. However, it seems justified to claim that the introduction level characteristic probably had indirect influence on the study outcomes by increasing the ease of use and readiness-to-use of the BaRE method. Further, this focus has probably also influenced the relative part of the relative advantage characteristics by suggesting feasible and compatible rather than optimal practices. This is exemplified by a quote below in the description of the relative advantage characteristic (p. 158).

Relative Advantage. The essence of the relative advantage characteristic was that the new way of working should prove better than the previous way of working (p. 91). The basic assumption in the method development was that most companies lack basic RE practices. Therefore, the goal of focusing on introducing basic practices like documenting and a systematic way of working should represent a fair but manageable improvement for the anticipated target adopter group and, consequently, the method should be beneficial for them.

There were two common actions in all the three companies, the first of which was learning more about RE. This was achieved by personal studies, participating in training, and spending time with the author of this thesis; overall, this action resulted in increased understanding of RE (Table 32, p. 135). The other common action was the adoption or development of a new common RD template. In companies A and C, old templates were abandoned and a new modified template replaced them – even if C chose to maintain two physically separate templates due to two competing text processors used in the company. In these companies, the establishment of a single RD template resulted in a common context for the RE work and this was reported to have made it easier to communicate about RE with colleagues. Namely, previously used different templates had also competing definitions for some terms and the unification finished disputes about terms and their definitions.

Companies A and B, on the other hand, adopted RM tools and developed a standard change management process at the same time. In Company B this process was also adopted and by the end of the study the new process was reported as a real standard way of handing change in the company. As a side effect of this tool adoption, both of these companies reported satisfaction with the fact that now there was a standard place for requirements so that they could be located after the development phase. As company specific improvements Company A reported that the standard requirements development process provided the needed visibility for project progress and made overall project management easier. Company C, on the other hand, adopted individual practices from the *Practical RE in Short* guide from which the CRUD operations documenting for each use case was a prime example. Thus Company C saw relative advantage in the suggested RD template and training material and, consequently, they adopted the most useful features for the company. In the same vein it can be estimated that the suggested requirements development and management practices in general did not seem to provide advantage enough to invest time in adopting them. This estimate was supported by a general lack of time for improvements throughout the project and, on the other hand, fairly competent personnel in the company (Table 31, p. 134). Company B reported numerous improvements including documented requirements and change requests. The key benefits from this

documentation approach were that now it became possible to plan future releases based on real requests rather than rumors and beliefs; similar requirements could be identified early on after their invention so that minimal time was lost analyzing them; and finally, having a standard electronic location for requirements provided assurance that all the requirements and change requests were processed and managed in a systematic and standard manner. Overall, these changes were improvements that solved individual problems, made working easier, and provided the companies a relative advantage in comparison with their previous practices. The IT manager in Company A could foresee this situation already in the initial state interview before they even had the final BaRE method documentation since he explained the desire to adopt the BaRE method in the following way:

[The BaRE method is] Definitely better than anything we are currently using. Or if you consider, for example, the RUP – it takes so much time to learn RUP that we just cannot adopt it here. So the BaRE method is on the right level with respect to our needs.

The Company C managing director, on the other hand, reported that they wanted to adopt the BaRE method since it seemed to provide improvements in many areas:

One reason for adopting the BaRE method as such is that we do not have anything ready yet. So if we adopt it as a whole, we shall immediately get improved practices in many areas.

It is evident that the participating companies evaluated the relative advantage of adopting the BaRE method and made their decision based on the available facts and perceptions. In the case of Company C, it was also observed that the lack of the relative advantage in some areas seemed to have limited the scope of adoption. Thus the relative advantage is estimated to have had direct influence on the outcomes of the present study.

Compatibility. Compatibility was earlier defined to refer to consistency between existing values and past experiences (p. 91) but in a practical effort the compatibility with current practices and equipment are equally important. Since such a comparison is not possible with only a method, two other aspects are considered first: first, the method is expected to be compatible with the potentially known general software engineering practices and, second, with the potentially known RE practices. Considering actual practices, it is assumed that few people in companies with a low maturity level are familiar with formal methods in general or novel approaches to problem solving like the problem frames (Jackson 2001). Thus the compatibility is taken to refer to natural language and informal specification techniques possibly complemented with basic diagrams or semiformal techniques. Consequently, the suggested document template is built on natural language descriptions, the use of multiple attributes for each requirement, and the development of four basic diagrams of the system: the context diagram, the business process diagram, the data model, and the dialog map. Since the suggested document templates and practices are in most cases adopted or adapted from the basic level RE literature, it is assumed that they provide some degree of compatibility with any experience in the RE area.

In the course of the present project, the compatibility issue was raised with all the companies at some point of time. In Company A, halfway through the second phase it became apparent that

there was some resistance to start using the new practices. The IT manager reported this problem and mentioned that people had started missing some of the aspects they were accustomed to having in previous RE assignments. However, due to the late appearance of this issue it was not possible to see how it was concluded; the fact that the pilot project started providing evidence of the benefits of the BaRE method was a strong rival for the compatibility issue. The IT manager also reflected on this topic and estimated that the BaRE method would be much easier to adopt in a company without any previous practices.

In Company B, no compatibility issues were reported with respect to the requirements development phase but the new change management process experienced difficulties before it was really adopted in the company. The new process was based on the *sfrm* tool into which the salespeople were expected to enter new requirements and change requests; the old practice, on the other hand, was to drop a paper form in a mailbox or in the developers' room. In spite of the training and initial support provided to all the employees it was reported to have been quite hard to get this change accepted, and after the development personnel changed the salespeople slipped back into the old practices. However, in both the cases some support from the managing director and a determination not to accept requirements or changes on paper finally produced the desired outcome and people started following the new process.

Since Company C focused on the RD template development, some compatibility issues were also identified in this area and further incompatibility issues were found in the requirements management tool area. For example, the software architect reflected on the other RD templates he had seen before in his career and noted that the BaRE template included things he had not seen before:

And then there was this system description, first I thought that gee – do I need to describe this here? In all other documents I had done before I had described the system through requirements and now here I had to separate the system and requirements. So first I needed to describe the system and the things the system includes; going into details only thereafter was quite hard for me in the beginning. This, however, is due to the fact that I have been working with the old fashioned templates before.

Also project manager C3 reported to have confronted new approaches, like putting the glossary into an appendix instead of in the beginning of the document, which had slowed down the adoption process. In the same vein as the software architect, he had also found it best just to accept these new conventions and had started living with them.

Company C did not adopt any requirements management tool during the present study and a discussion on the reasons for this brought up two constraints halfway through Phase 2. The first of them was incompatibility with existing hardware and software which was explained by the software architect as follows:

We did not adopt the *sfrm* since the people who would use it are using Windows NT and thus the tcl/tk –toolkit or X-Windows environment should be installed. ... And even though their installation would be easy, we need to have a shared data store for the requirements – it is quite an unacceptable approach to store requirements in unknown locations in individual computers. That is simply not acceptable in our company.

The basic requirements for an acceptable solution were summarized as running on both Microsoft Windows and Linux –environments, preferably with a web-interface, and having a shared data store. This compatibility issue was so central that even though, a year earlier, the company was offered the possibility to continue using one of the leading configuration and change management systems without license costs or alike, they did not do so since they would have needed to set up and maintain a Windows-NT server for it. This leads to the second constraint in Company C which was that they did not have time to search for suitable solutions, but somebody should come up to them and show a solution that would be compatible with their environment and, of course, also address other topics that are important in the adoption process.

As the previous examples demonstrate, the BaRE method presented some clear incompatibilities with previous experiences. However, it should be kept in mind that in the beginning of the project the RE practices in the companies were in initial level (p. 152) and all companies had emphasized the need to improve their practices (Section 8.2.1, p. 132-). Thus a discontinuity between previous and new practices is only natural. Further, even if the examples above demonstrated incompatibilities, the BaRE method was not only incompatible. For example, companies A and C had documented requirements already before and all participants had used techniques like interviewing, introspection, and meetings before.

The compatibility issue was encountered multiple times in the present project. In the initial interview, compatibility did not raise concerns but in the actual implementation phase it became a clear factor in decision making. In Company A the new practices raised some resistance in actual work, in Company B the new change management process required extra attention before the salespeople adopted it, and in Company C the new RD template structure raised resistance before acceptance. The clearest sign of the compatibility issue, however, was the fact that Company C rejected adoption of the suggested requirements management tool due to its incompatibility with the company values. It should be noticed, however, that numerous aspects of the BaRE method appeared compatible with the adopting companies and people – e.g. use of standard text processing tools, basic RE techniques, etc. Thus it can be concluded that the compatibility had a direct influence on the outcomes of the present study.

Ease of use. The ease of use was earlier defined to refer to the degree to which a person believes that using a particular system would be free of effort covering also the learning aspect of the system use (p. 91). There are three key issues that are expected to make the BaRE method easy to learn and use: focusing to RE only, providing detailed guidance for adopters, and making the method descriptive. There are many clear needs to expand the method and integrate it with, for example, project management and architecture design but in the present work these topics were left outside the scope for simplicity and ease of use. In fact, the method has even now an extensive coverage since it spans the whole RE area and not just, e.g., the requirements development phase. The method does provide varying degrees of details on completing the suggested tasks; the most detailed descriptions are provided for the contents of the requirements document and how to integrate it with the requirements development process. Finally, the method includes no control over whether an adopter actually follows the provided guidance or not. Thus, even though the method includes sufficient detail to be used as a prescription for doing RE, it is still intended to serve as a comprehensive description on how RE can be done in a limited application domain.

In practice, the ease of use proved an interesting characteristic. The basic problem is that the level of detail that makes it possible to use a method by following the provided instructions

often leads to a complex solution. In the beginning of the project, all the people considered the suggested method easy to use but after getting some first hand experiences of it, the answers became twofold – all the people seemed to agree that, basically, the method looked simple but actual use was not easy. A quote from the Company B senior software engineer represents all the replies to this question well – except that some started with “No, it isn’t”:

Yes, it is. The biggest initial problems were caused by uncertainty about how to actually complete the document – how should you locate different kind of requirements in it? But once you got over this problem, it was easy to use. It did not really take too long to see how to fill in the document. And the fact that the documentation actually consists of multiple documents made things even easier.

In the same vein the replies to the question whether additional documentation should be provided were split in two – some found the existing documentation sufficient and some insufficient. However, a need for more examples and especially for a complete example RD for a single system seemed evident. Overall, the documentation was found understandable, logical, and comprehensive with one exception – the use of English language in the documentation was reported to have complicated it to some extent. Even if the method documentation was found to have some important improvement opportunities, one should also bear in mind that only two of the participants used three working days for studying RE, two used two days, and four managed with less than half a day of study. In this account the Company A IT manager was not included since his duties included corporation level activities making deeper study of topics necessary and, on the other hand, most of these people used much more time on RE development work which could, of course, also include studying.

Acknowledging that the adoption of new practices requires some degree of learning, it seems that the BaRE method is fairly easy to use. A key factor to this end can be expected to be the fact that the method was descriptive and no company followed it to the letter. A major step in making it even easier to use, on the other hand, would be making it available in the local language. Overall, it appears justified to claim that the ease of use had a direct influence on the study outcomes.

Trialability. Trialability was earlier defined as “the degree to which an innovation may be experimented with on a limited basis” (Rogers 1995, p. 16) and it was also noted that the ability to try out an idea on an installment plan can expedite the adoption (p. 92). In the BaRE method the focus is on providing easy project based trial use rather than trialability of the method’s individual parts. The general idea is that the method should be so ready-to-use, easy to use, compatible, and comprehensive that there is no need to invest in adaptations or learning before trial use. Instead, a potential adopter should be able to try the method as it is on a project by project basis, and decide about possible adaptations and the future thereafter. However, the clearly defined key components of the method (Section 6.1, p. 98) make also the development of customized installment plans feasible.

In the present project, the need for trialability as such was not encountered but all the participating companies expressed a desire to try out the method as it was before modifying it to actual needs (see also the quotes below for the Comprehensive –characteristic). As this aspect refers to a kind of trial use, these two aspects are discussed below under the readiness-to-use characteristic. The installment plan approach was also evident in the participating companies. For example, Company B reorganized the requirements management area first and only

thereafter moved to requirements development practices. Company C, on the other hand, focused only on the requirements document template and training in the course of the reported study. Thus there is evidence that the BaRE method can be adopted on a limited basis and also be claimed as trialable. So, even though the BaRE method was not developed for *experimentation on a limited basis*, adopting the method based on an installment approach proved important in practice. In short, it can be concluded that trialability had direct influence on the study outcomes.

Result Demonstrability. Result demonstrability was earlier defined to refer to the degree to which the results of using the innovation are visible to others (p. 92). Examples of potential results of the method adoption include a decreased number of change requests, a decreased number of defects, or increased user satisfaction. However, these kinds of outcomes are clearly outside the method design scope and can be observed only in practice. Nearly the only concrete results that can be claimed to be produced by the method by its design are a requirements document and documented change requests – especially if the company has not produced them before.

The present study resulted only in anecdotal evidence of the benefits of the BaRE method adoption. In Company A the system analyst summarized her experiences in short as follows:

The requirements document has been useful and we could not have even started this project without it.

No other companies made such clear statements of the benefits of the method use. However, in Company B their first requirements document was developed and in Company C four requirements documents were developed based on the suggested template. Furthermore, all the three companies in the present study mentioned a low number of changes in the conducted projects. The Company A system analyst had the most accurate account on this aspect:

In my opinion it is worthwhile to work this way – if you do the requirements document carefully from the beginning you will avoid changes in the end of the project. As a matter of a fact, we have made really few changes in this project. When we started the implementation phase we had 57 tasks to complete and today there were 58 of them – some tasks have been removed and some new have been added, of course. But in my opinion this project has had really little changes.

Discussions on the reasons leading to a low change rate in this particular project revealed that the project team had previous experience with the system and the schedule had been looser than normally. However, the investment in the requirements phase was still found an important contributor to the prevailing situation where no real bad surprises had come up and the project went surprisingly well despite the initial difficulties. Overall, the project was reported to be well under control from the project management point of view.

In the end of the present project Company B declared one major benefit from the documented requirements in addition to the above-mentioned low number of problems or changes in the developed requirements document. This benefit was the ability to combine new requirements with previously recorded similar ones which was stated to result in a fair reduction of workload.

In the end of Phase 1 the software engineer demonstrated the result of the adopted change management process with the following statement:

As of now change requests have been well under control. And nobody comes in any more saying that “but we need to have this done *now*.”

In Company C, the project managers for the two projects in Phase 2 noted that not many changes took place in their projects. Further, in the project C-P3 the customer had read the produced requirements document and claimed to have understood it with one reading. This unusual positive feedback from a deliverable got a fair amount of attention in Company C – a more common situation was encountered in project C-P4 where the project manager was happy since he “did not get feedback from or lashed by the customer.” In the small project C-P1 the software architect estimated that he would probably have managed to develop the software without a requirements document but it had helped to set boundaries to the solution and, on the other hand, it had served as a checklist in the implementation phase. Thus he concluded that the use of a requirements document had surely speeded up the development phase and that he had actually *used* the document after it was completed. A final point about the benefits of the BaRE method use in Company C was brought up by the project manager C2 who stated the following:

I have seen that even between me and the developers we need to have this glossary so that we speak the same language. ... Well, quite in the beginning I did not have it written down even if the heading was here, but we soon realized that we needed to have it fixed before we could work on the project efficiently.

Tracing the document history in the version management system revealed that the first requirements document after the project initiation was followed by another one with a fair glossary only three days later.

In the beginning of the present study no experiences from the suggested method in practice existed. Thus, in the present study the result demonstrability did not have any influence on the initial method adoption decisions. However, in Companies B and C with the installment adoption approach it was evident that the gained positive experiences from the adopted practices had a direct influence on further improvement of the practices.

Visibility. Visibility was earlier defined to refer to the degree to which innovation adopters can see the innovation itself (p. 92). Visibility is a clear problem for a method that is by its very nature not tangible or visible, but some subtle decisions and actions were made to increase the visibility and concreteness of the suggested approach. The first such decisions were to call the approach a method and to tag it with a name, which makes it possible to refer to the approach with a simple and unique way. Other respective decisions were the creation of method documentation and publishing it as a bound report with covers (Nikula 2002b). The two last examples of the visibility of the method are the explicit linkage between the requirements document template and the requirements development process, and the numerous supporting document templates that have the same general structure and information derived from a national standard. Thus a fair attempt was made to make the developed method as concrete and visible as possible even if the possibilities for it are limited.

Since the actual method documentation was completed only after the initial state interviews of the evaluation project, the visibility of the method was minimal at that point. However, in the course of the project also the visibility of the method increased. The most visible change in Company A was the adoption of a requirements management tool which, consequently, became associated with the new way of working. It was also decided that in Company A only one requirements document template would be used in the future, and the BaRE guide was supplemented with two documents describing the company specific adaptations of the practices, one for requirements development and another one for change management practices. By the end of this project only one requirements document based on the new template was developed, but it provided an example of the new approach anyway. In Company B the most visible things were the adoption of a new tool for change management and the development the first requirements document in the company. Other important changes in Company B were the production of reports on the suggested changes and their status, guides on the change management practices, and documented change requests. In Company C four requirements documents were produced during the present study with a clear direction to a unified and BaRE like template (Table 37, p. 140). Otherwise, basically the only other change in visibility in Company C was the revision of other document templates to follow the structure and contents of the new requirements documents. The most evident observation concerning visibility in Company C was the way the bound BaRE documentation was adopted in the company – after the project managers had studied it for a month or so, another copy of the document was requested. Namely, the original one was read and marked so heavily that it was considered inappropriate to show it to customers as documentation on the company's way of working.

It is clear that the method visibility did not have any influence on the initial adoption decisions in the present study. However, as the project went on, the visibility of the method and especially the way the companies actually implemented it in their own organizations became more evident and visible. Thus, even though in the present study visibility did not appear important from the project outcomes point of view, the characteristic itself seemed important in the companies.

Comprehensiveness. Comprehensiveness was earlier defined to mean that a method should address all the key areas in RE: elicitation, analysis, documentation, validation, and management (p. 93). The BaRE method suggests techniques for all these areas and processes for both requirements development and management (cf. Figure 24, p. 110). The *Practical RE in Short* guide names numerous additional techniques for all these areas (see Appendix 6. RE Practice and Technique Summary). Thus it provides a more comprehensive coverage of RE areas than most other found methods (Section 7.5, p. 121). However, the depth of the handling does leave room for enhancements.

Bearing in mind that in the present study no clear distinction was made between requirements management and change management (p. 29), all the participating companies claimed to have carried out some activities in the above-mentioned five areas in the projects followed with two exceptions. In project A-P1 the change management process was not yet needed and otherwise the new defined process was not yet used in the company; in Company C only project C-P4 reported explicit analysis activities.

In the requirements development area, all the companies adopted new or revised requirements document templates and new documents were developed based on them. There is no such direct evidence from conducting elicitation, analysis, and validation activities, but looking at the techniques used in the projects provides some information about these activities (Table 40).

Further support on conducting activities is provided by the developed use case documents, review meeting records, and requirements documentation. In Table 40, companies A and B are represented only by their single projects while the Company C column summarizes the practices as they were done in the company by and large. In Company C all the practices were used at least in three of the four projects with the following exceptions: in Project C-P1 clearly a smaller set of techniques was used due to the small and different kind of nature of the project; two-level documentation was done only in the last two projects; Project C-P2 did not develop any diagrams, and C-P4 did not develop any prototypes. Notice also that Company C did not claim use of CRUD matrix but they started documenting these operations for each use case in their requirements documents.

Table 40. Suggested techniques for requirements development phase and how companies used them

Techniques	Company A	Company B	Company C
Apprenticing			
Checklists			
CRUD Matrix Development			
Define System Boundaries	X	X	X
Define the System's Operating Environment	X	X	X
Diagram Development		X	X
Document Studies			
Electronic requirements		X	X
Interviews		X	X
Introspection	X	X	X
Meetings	X	X	X
Prototyping		X	X
Reading	X	X	X
Reuse Requirements	X	X	
Reviews	X	X	X
Scenario Development		X	X
Two-Level Approach to Requirements		X	X
Documenting			
Write User Manual			

In the requirements management area companies A and B adopted new tools and defined new processes. In Company B these processes were also adopted in use while in A this phase was still in progress. In companies A and B the change requests were managed in a systematic way in the end of the present study and all the companies used basic techniques for management as baselining, versions identification, and change history maintenance for requirements documents and unique identifiers for requirements. The fact that Company C had not paid much attention to the management area was brought up in the SPICE assessment (p. 142) and in the discussions with the company representatives.

In the discussions with the participating companies, links to project management, architecture, and metrics were brought up. However, within the RE area no omissions in the method were identified. In the initial interviews with the companies, the reasons for planning to adopt the BaRE method as such were explained in fairly consistent ways. For example, the managing director for Company C explained this desire in the following way:

So if we adopt it [the BaRE Method] as a whole, we shall immediately get improved practices in many areas. If we had something available for some area, we could benchmark the method with our own practices and we would do it – but as of now we have only some ideas how to do things and none of them is as advanced as the BaRE method.

The IT manager in Company A went into more detail in explaining his interest in the BaRE method:

[The BaRE method] provides a simple and understandable model of how to do requirements engineering. It provides a model of how to do things, it has been thought through so it seems justified to try it out as such and see if something needs adaptations. As a whole it seems very logical and no part of it appeared clearly unfeasible but everything seemed quite doable.

Towards the end of Phase 1 the project manager C2 explained the benefit of a comprehensive document template in the following way:

So if you do not want to adopt the template as such, it is easier to start developing a new one from this since it has been thought through before, and there is little need to add something in it. Rather, you can possibly take something away from it.

In the present study the adopting companies reviewed the available material and after confirming that it addressed the problems they had been looking at, they were ready to make adoption decisions from the technical point of view. In the course of the study, the comprehensiveness of the BaRE method was re-evaluated to confirm that the participating companies did not perceive any key area in RE to be missing from it. Since two of the companies addressed all the suggested areas, it seems clear that the comprehensiveness had a direct influence on the project outcomes.

Readiness-to-Use. The ready-to-use characteristic was earlier defined to mean that a method can be taken into use without adaptations (p. 93). This characteristic was addressed by developing the RD template and the requirements development process as synchronized elements and by selecting requirements development practices so that adaptations become unnecessary before use in the target application domain. In the requirements management area, the respective elements do not appear quite as ready-to-use even if the practices and processes should prove useful for adopters. The training part of the method is limited to documentation, which leaves a great deal of room for improvement, but for an individual gaining access to the method documentation this provides a straightforward way to start learning the method. Finally, the tool support for requirements management is limited to document templates for a word processor. This again can be considered only as the first step in the tool support area but,

however, the templates do provide a quick way to start developing requirements towards the final requirements document.

In Company A the BaRE method was studied in the beginning of Phase 1 at the same time the participants were acquainted with the requirements management tool vendors. The Telelogic's DOORS RM tool was officially selected as the company tool in the middle of Phase 1 and soon thereafter Telelogic arranged a two day course in adopting the tool for six people. Only two weeks later, a three day workshop on requirements development was held by Telelogic for six people. One part of this workshop was to discuss and accept some changes in the BaRE requirements development process in order to synchronize it with the tool; the changes were suggested by a consultant. After the changes were accepted in the workshop, the consultant implemented the necessary changes in the tool and delivered them to the company making the tool ready for use; the process and changes to it were documented in a 19 page slide show. Again, two weeks later half a day of training in requirements writing was provided by the same consultant for 21 people. These infrastructure establishment activities for the requirements development phase were implemented in less than two months after the actual tool purchase decision.

In Company A, change requests and bug reports had been handled earlier by helpdesk software. In the course of acquiring the new tool it was decided to start using it for handling change requests in-house and a new document called "Defect reporting and change request process" was developed including 26 pages and published three weeks after acquiring the tool. The basic approach was to continue using the existing helpdesk system in customer support for recording customer problem reports. After analyzing the reports in the helpdesk system, the ones resulting in changes in software implementation were transferred to the change management module of the requirements management tool. Thus the main change on the change management side was to document the changes and adopt a tool to help the development team to manage changes. The new infrastructure and document for change management were ready within a month after the tool purchase.

Company B started working on their change management process right after the initial interview of Phase 1. In two months, the *sfrm* tool had been adopted for change management and after another two months a user guide had been written and most of the people in the company had been trained in the tool use. Thus getting the infrastructure to work for change management had taken less than four months in Company B. At this point the interest in the requirements development process had increased, but due to the changes in personnel this topic was actually addressed only in Phase 2. Once the new senior software engineer started working on requirements development the actual process seemed quite fast and straightforward again. There were some initial problems in getting acquainted with the BaRE requirements document template but once these were discussed with the author of this thesis and the BaRE documentation had been walked through, the actual requirements development work was started with minimal adaptations to the template (p. 140). The actual requirements document was developed with the MS-Word text processor and later on people submitted new requirements with the *sfrm* tool. The first version of the requirements document was developed halfway through Phase 2 and at the end of it a completed and reviewed document was ready.

In Company C, the changes focused on the requirements document templates as summarized on pages 137-142. In short, the templates were developed first in Phase 1 and then revised in Phase 2; even though a lot of work seems to have been done on adaptations it must be noted that the

revised version looked a lot more like the BaRE RD template than the first version (Table 37, p. 140). At the end of Phase 2 this template was commented on in the following way by the project manager C2: “we still need to do one more common review of the template before we can call it a company standard template.” Thus a year working on the template did help improve the situation in the company, but did not result in a standard template yet.

In the beginning of the whole BaRE method evaluation project the readiness-to-use was stated an important characteristic by the companies A and C. The managing director for Company C put it clearly in his following statement:

Well, it is not an end in itself to modify the method. My guess is that the start-up will take place very much on an as-is basis. Namely, you can expect even the very first use to reveal that in the given case it would be better to do things differently or we find it better to do it like this etc. But that's fine – the evident overkills, or how ever you call them, are very easy to detect once you start to work with it. So we are surely going to look at those things closer but it is really not an end in itself to modify the method. If you think about how we work in general as a software house, we really are not that different from all the other software houses.

The software architect supplemented this statement with a note that “I think we really do not have too much time to invest in adaptations.” Bearing in mind that companies B and C used only about 2 person weeks each in requirements development activities during the 11 month long project and Company A managed with less than 2.5 person months (Table 39, p. 145), it seems clear that no extensive adaptations were done during this project in these three companies.

The role of readiness-to-use was discussed both in the beginning of and during the project but this characteristic was never directly considered important. However, a desire to adopt the method without adaptations to avoid spending extra time on adoption before the method could be used in practice was reported. Furthermore, considering the data discussed above for the Comprehensive –characteristic, a thought through approach seemed valuable for the participating companies. Supplementing this thought through aspect with the observed time use in the present project, there seems to be evidence to support a claim that readiness-to-use had, after all, direct influence on the study outcomes.

Adaptability. The adaptable characteristic was earlier defined to mean that a method should allow for adaptations and support them rather than constrain them (p. 94). The BaRE method proposes some ideas on customizing the method to different needs. The most important way to adapt the method is the possibility to adopt individual parts of it in practice. The most likely elements to be adopted individually are assumed to be the requirements document template, change management process, and some individual techniques. In short, the method can be adapted in any way that is seen fit even though claiming to follow the BaRE method could be expected to mean following the major elements of the method by and large.

The BaRE method was implemented from five major components: a requirements document template, requirements development practices, requirements management practices, tool support for requirements management, and training. Next, the ways participating companies adapted these components to fit their needs are summarized. Company A developed

requirements document templates both in Finnish and English. The English template contents and headings were very similar with the BaRE template (Table 41) but the Finnish template deviated quite much in the terminology used. This situation seemed very similar to the terminology deviations in Company C (p. 142) even though the terms did not match with them, either. The differences in the headings were not very radical: Company A had some additional subheadings for customer problems, document overview, and references but omitted the whole introduction chapter and reserved one chapter for each use case. The most important reason for the template revision was to integrate it with the tool implementation. This meant, for example, splitting the document into six different sections for easier reuse between different projects. Notice that Table 41 uses the same term definitions as Table 37 (p. 140).

Table 41. Company A and the BaRE requirements document template comparison. The BaRE template has 51 headings; the Company A Finnish template includes also use case templates with 20 properties for each use case

Company A RD Templates		English	Finnish
<i>General</i>	Identical	0	0
<i>Properties</i>	Similar	0	0
	Partly similar	1	1
	Different	2	2
	Added	0	0
	Omitted	6	6
	Count	3	3
<i>Document Headings</i>	Identical	47	12
	Similar	2	28
	Different	0	7
	Added	6	5
	Use case names	0	0
	Omitted	2	4
	Count	55	52

The requirements development process was modified so that the first five steps in the process (Table 18, p. 108) were repeated until the requirements were found complete. After this, requirements prioritization and document completion, analysis, and validation were to be done under the change management module of the selected tool to allocate requirements to releases and finalize the document for a specific release. The slides documenting these adaptations provided also a great deal of concrete advice how to use the selected tool to accomplish these tasks. Some further details from different viewpoints on the requirements development process are outlined above in the Comprehensiveness (p. 164) and Readiness-to-Use (p. 166) characteristics. On the change management side the most different aspect was the fact that Company A used two tools for it – helpdesk software as a front end from the customer side while the new tool was to be used internally by the development people. Otherwise not much more was done in addition to documenting the process, tools, and how the system was to be used. Thus a company specific solution to change management was developed even though it was built mostly on the basic practices suggested in the BaRE method. As it happens, these

kinds of tasks suit the tools very well, and thus, Company A was not satisfied with the text processor approach suggested in the basic BaRE method. Company A was also the only company to buy external training to complement the BaRE method documentation (p. 146). Based on the discussions in the course of the evaluation project, no other people read this documentation but the IT manager and system analyst who participated regularly in the meetings with the author of this thesis. Thus Company A adopted the approach suggested in the BaRE method but adapted all its components to the company needs.

Company B made only marginal adaptations to the requirements document template (Table 37, p. 140) and also requirements development practices were followed in relevant parts (Table 40, p. 165). Considering the change management practices, the company adopted a tool to support the task and implemented it in a way that seemed to suit the company well even though the requirement document development was done with a text processor. In the training area no training was arranged but some time was spent studying the BaRE documentation and, on the other hand, discussing with the author of this thesis (Table 39, p. 145). Thus, again, Company B followed the approach suggested by the BaRE method but made some adaptations to all components of it except for training; especially no attempt was made to follow the method in detail but only in principle.

Company C developed two versions of their requirements document template (Table 37, p. 140) so adapting the template seemed important in the company even though the later template did not deviate radically from the BaRE template. However, it was clear from the beginning that nobody in this company wanted to use tables for documenting the requirements since it was found impractical with large documents crossing page boundaries. Another interesting practicality was that two separate document templates were needed in the company – one for Microsoft-Word and another one for the LaTeX text processing system. Company C people paid little attention to the requirements development practices, but they still used quite a few of the techniques suggested in the method (Table 40, p. 165) and started documenting the CRUD operations for each use case. On the training side, the electronic version of the BaRE documentation was not received very well, and only after the bound version of the documentation was delivered the method study seemed to become easier. An option to arrange a training day for all the related people was also discussed in length in the beginning of Phase 2, but no concrete actions followed from this discussion. As an overall estimate, Company C did adapt the method to some extent with clearly the biggest adaptation being to adopt only two of the suggested five components so far: the requirements document template and the training material.

Overall, the adaptability was a very important characteristic for all the companies participating in the BaRE evaluation project. The descriptions above of the adaptations done in the selected approach provide clear evidence that the method is actually adaptable and further support can be found in the quotes of the managing director for the Readiness-to-Use (p. 166) and project manager C1 for the Comprehensiveness (p. 164) characteristic explaining their approaches to method or document template adoption. Thus it is concluded that the adaptability had a direct influence on the outcomes of the present study.

Summary. Based on the conducted analysis, the BaRE method implementation addressed all the expected characteristics in varying degrees. Furthermore, the empirical evaluation supports this view. Table 42 summarizes how the BaRE method was perceived to match the needs of the companies in the course of the reported case studies. The symbols in the table are interpreted as

Table 42. The BaRE method characteristics and the perceived match in the conducted case studies in the course of the evaluation phase

The BaRE Method Characteristics	Company A	Company B	Company C
3-Tier Architecture Focus	●	○	●
Small Application Focus	-	●	●
Small Team Focus	●	●	●
Introduction Level Focus	○	●	●
Relative Advantage	●	●	○
Compatibility	○	○	○
Ease of use	○	○	○
Trialability	-	○	○
Result Demonstrability	-	-	-
Visibility	-	-	-
Comprehensiveness	●	●	○
Readiness-to-Use	●	●	○
Adaptability	●	●	●

follows: character ‘●’ means *Complete match*, character ‘○’ means *Partial match*, and character ‘-’ means *No match*.

Table 42 can be summarized as follows. The BaRE method focus matched the needs of the participating companies well. The 3-tier architecture focus matched the application type of Companies A and C, but with Company B’s system administration tool the method provided only a partial match. All the developed applications were small ones developed in less than a year with less than a person year of effort; since the Company A project did not concern a normal software application development project it is marked with a dash in the table. All the conducted case studies were implemented with small teams with at most 4 people. The introduction level focus reflects the different levels of expertise the participating companies had in RE: Company A’s decision to use a commercial requirements management tool and the involvement of numerous people in writing requirements suggests that it was not clearly an introduction level company in RE; Company B utilized pretty much all the components of the BaRE method; and Company C focused on requirements documents and training but also indicated interest in requirements management and tool support for it.

In the diffusion of innovation characteristics the BaRE method did not do so well. In Companies A and B the method provided relative advantage but in Company C only partial relative advantage was identified. The compatibility issue in general is a difficult one and the method can be claimed to have provided this only partially in all the participating companies. In the same vein, ease of use was reported at first sight but after actual work with the method such claims became rare; thus only partial success in this characteristic is claimed. The trialability was not present as such but since companies B and C clearly adopted individual parts of the method in different times, it can be claimed that the method supports this characteristic. In Company A all the changes were basically implemented at the same time, so no trialability was utilized in this case. The evaluation project was initiated without any evidence of benefits

resulting from the use of the method and without visibility since the method was not yet completed at the time of the initial adoption decision. The result demonstrability can be expected to have had a role in reinforcing the continued and increased adoption of the method in Companies B and C with the installment approach, but in the lack of tangible evidence to support such claims no match with the needs is claimed. In the course of the study the visibility increased especially with the publication of the bound method documentation, but even though it was received well by the companies, there is little evidence to support a claim that it would have been needed in the present study.

The third group of characteristics that were novel in the present study seemed to match the companies' needs well. Companies A and B implemented changes in all the suggested areas but Company C focused only on two of these areas; however, the comprehensiveness characteristic was perceived important in all three case studies. In the same vein all the companies wanted the changes to take place with minimal adaptations, and evidence for the importance of the readiness-to-use characteristic was found. Company C modified their own template from the suggested one so they did not perceive it initially ready-to-use even though their second template made a tangible movement closer to the original BaRE template. It appears evident that there is room for improvement in this characteristic. Finally, all the companies made more or less modifications to the provided artifacts; so clearly, adaptability was a central characteristic in the present study.

8.3.5. Factors Affecting the Case Study Outcomes

The conducted empirical evaluation identified a number of factors that affected the outcomes of the present study. The used evaluation framework approached technological decision making from three different viewpoints: environmental and organizational contexts and technological innovation (Figure 28, p. 148). In the environmental context, only external information sources and business partners were found as potential contributors to the outcomes of the present effort. Specifically, these factors were deemed to be the RM tool consultant who implemented the BaRE method and RM tool in Company A, and the author of this thesis. In the organizational context, the key factors were identified to be management awareness and attitude towards the improvement effort, company size, and employee skills. However, since company size and employee skills were not found central from the organizational point of view, they are addressed below as the technological innovation characteristics. Thus the organizational factors are limited to the existence of sponsors and champions in the present study, and the technological context is approached from the method characteristics point of view.

The RM tool consultant cannot be claimed a central player in the present effort since he operated with only one of the companies and, on the other hand, there are numerous other consultants in the same company as well as other tool vendors. However, from the point of view of Company A, the effect of this consultant on the study outcomes was a clear one since another vendor was not found capable enough to warrant for making the tool purchase decision. Based on the fact that the tool was found crucial in Company A, the tool vendor and the consultant, therefore, have also had an effect on the study outcomes. Estimating the effect of the author of this thesis on the study outcomes is also a complicated one. As is reported above, the company representatives were skeptical whether anything concrete would have happened without the presence of the author, and at the minimum changes in the extent, schedule, and level of detail were estimated to have occurred. Whether a consultant could have taken this role was partially

reflected on by the IT manger in Company A when he stated that an academic approach provides additional value for the work. In an earlier discussion he also noted that he expected both industrial experience and postgraduate level knowledge from a partner in this process improvement effort. Thus, even though this topic could not be studied in a clearly objective manner, it seems that in the present study the author of this thesis had an important role on the outcomes of the project.

In the organizational context the role of the management is likely to have played a central role in the present study. In Company A the actual sponsor of the effort was never encountered, but in the small companies the managing directors clearly adopted this role. On the other hand, the champion in Company A was the IT manager while in Company C this role was clearly possessed by the software architect; in Company B no clear champion was identified due to the small number of people involved in the project. It seems evident that if any of the people named had been missing in the present effort, the outcomes of the project would have been very different.

Table 43. BaRE method characteristics and their influence on the outcomes of the present study

The BaRE Method Characteristics	Direct Influence	Indirect Influence	No Influence
3-Tier Architecture Focus		X	
Small Application Focus		X	
Small Team Focus		X	
Introduction Level Focus		X	
Relative Advantage	X		
Compatibility	X		
Ease of use	X		
Trialability	X		
Result Demonstrability			X
Visibility			X
Comprehensiveness	X		
Readiness-to-Use	X		
Adaptability	X		

Table 43 summarizes the BaRE method characteristics and their influence on the outcomes of the present study as they were identified in the conducted empirical evaluation. Table 43 shows that all the characteristics providing additional focus on the solution were deemed to have only an indirect influence on the present study mainly through the ease of use and readiness-to-use characteristics. Furthermore, the small team focus made it possible to increase the compatibility of the developed solution while the introduction focus allowed for increased relative advantage. The trialability as such was not found very important in the present study, but considering also the installment approach as trialability this characteristic had direct influence on the study outcomes. Finally, the result demonstrability and visibility were estimated to have had no influence in the initial phase of the evaluation project, but as these aspects became more

apparent in the course of the project it can be estimated that they had some influence in the later phases of the project in the companies following the installment approach.

Considering the above analysis it can be concluded that all the studied contexts affected the outcomes of the present study. In the environmental context the external information sources, the RM consultant and the author of this thesis, and in the organizational context the project sponsors and champions, i.e., the management involvement, were found important factors in the study. From the technological innovation point of view the relative advantage, compatibility, ease of use, trialability, comprehensiveness, readiness-to-use, and adaptability characteristics were identified to have had a direct influence on the outcomes of the present study.

8.4. Discussion and Conclusion

In practice, the different characteristics are not as clearly separated as described in Section 8.3. Instead, in many cases they overlap, support some characteristics and contradict others. In this section the most important interactions between the characteristics observed in the present study – relative advantage, compatibility, ease of use, trialability, comprehensiveness, readiness-to-use, and adaptability – are reported. After discussing these characteristics, some observations from the BaRE method adoption and innovation decision processes are reported, and this section is closed with reflections on the characteristics problems in software process improvement efforts.

Characteristics. The relative advantage characteristic was closely related with the readiness-to-use, ease of use, focus related, and comprehensiveness characteristics. Namely, the managing director in Company C clearly stated that modifying a method had no value in itself and they would rather just start using a method. This means that having a ready-to-use method provided them relative advantage since the adoption would be faster and cheaper for them. The IT manager in Company A, on the other hand, reported that the BaRE method is at the right level for them; in comparison with the RUP it requires less time to learn and is at the right level for their needs, which refers to the ease of use. The four characteristics specifying the focus on the method are also included to create relative advantage to more generic approaches since they give a head start in doing actual work by providing detailed instructions from the beginning. Finally, the comprehensiveness of the method makes it possible to see the whole solution in the beginning, and assess it for possible flaws and omissions even though the actual implementation of the method would be done in installments. Finally, the importance of the relative advantage as such is evident based on the present study. Namely, an analysis of the failure of Company C to adopt the requirements development and management practices suggests there was no perceived relative advantage in doing so.

The readiness-to-use characteristic had a close relationship with the ease of use, comprehensiveness, adaptability, and trialability characteristics. First of all, a ready-to-use method is likely to be perceived as easy to use and learn since detailed guidelines are provided. Second, a comprehensive approach provides assurance that all the relevant aspects have been considered since a potential adopter can check how the method solves topics that he/she knows to be problematic or difficult. Adaptability, then, is the backup for the case that some unforeseen problems occur as it allows for an experienced user to solve matters as he/she sees fit. Trialability is an interesting aspect that seems to overlap seriously with readiness-to-use. However, in the present study these two aspects have clearly different sets of mind. Trialability

is used to refer to the actions that are done without commitments or investments keeping in mind that it is possible to back off from the actions. For example, taking a trial run with a software tool at a vendor's shop involves no concrete investments but installing software in one's own computer and testing it with real data for hours or days is another thing. Backing up from such a trial run is still possible and not even very expensive. However, none of the companies in the present study acted this way. Instead, all the decisions were done based on the available information and every step thereafter was made to progress the process improvement effort. That is, if an action was not seen worthwhile it was not taken and, on the other hand, if it was seen worthwhile it was taken to get to the next decision point. In no case was the desire for trial use expressed. Thus the prevalent mind set in the participating companies was to adopt and use a method or artifact, and no desire to evaluate different alternatives with trial runs was brought up.

In the present study the compatibility characteristic was intimately related to the adaptability characteristic. Even though the implemented adaptations did not always seem well justified, the basic reason for adaptability is to be able to implement compatibility with existing practices and equipment. The managing director for Company C claimed that they are not so different from other software houses, but still, every company is different and thus the adaptability is often needed to achieve compatibility.

Adoption Process. One interesting observation in the participating companies was the way they implemented their adoption processes. There were three clearly different implementations of the BaRE method in practice: a limited method adoption scheme was taken by Company C, a quick and agile adoption process was implemented by Company B, and a full-scale implementation in one step was implemented by Company A. These processes are described in short in the following.

Company C adopted only the RD template from the BaRE method during the present study so it represents a limited adoption scheme. However, the initial plan was to adopt the method as such in one project, but as this project did not come true, the method adoption in full was abandoned. The next option thereafter was to improve the existing RD templates and for this purpose the BaRE RD template provided a promising benchmark. As explained above, the new template was modified quite a bit before people were happy with it, but even during this exercise the use of the template was steadily increasing in the company. The most prevalent characteristic of the taken approach is that improvements were implemented as part of production projects, and a working day or two were used for the improvement work in each project. All the improvements were immediately taken in use in production work. At the closure of the present project, hopes of extended use of the BaRE method were reported.

Company B seemed to follow an installment plan even though no clear vision was ever put forward about the RE improvement plans. Instead, small improvements were implemented and as the outcomes seemed promising, other improvements were planned and implemented in a similar manner. For example, after the first encouraging experiences with the RM tool and process adoption were gained, the interest in the requirements development process and RD template increased. Overall, this case provides the closest match with an installment approach where the full method was adopted in two phases in less than a year. The suggested method was basically adopted as far as possible in the form it was defined and only marginal adaptations were made when necessary. At the closure of the present project, the RE was reported to have

achieved the level of maturity that seemed sufficient for the company and further improvement efforts were planned in other areas in software engineering.

Company A was determined from the beginning to implement the BaRE method in full in one step. Parallel to the BaRE method selection also a requirements management tool was selected, and once the official company decision to purchase the tool was made, practically all the work on infrastructure was completed within a few months. This full scale adoption included training a large number of people in the RE area, adapting and integrating the BaRE method and its RD template with the selected tool, and the purchase of a sufficient number of licenses for full-scale use of the tool in the company. Only after the infrastructure side was completed was the first pilot project implemented with it. Overall, the planning and study phases of the improvement effort had taken over a year in the company before the final decisions were made, but thereafter the changes were implemented very quickly. During the later half of the present project the first experiences from the new infrastructure were gained. At the closure of the project the RE infrastructure was considered sufficient for the time being in the company and the only concern in the RE area was to get it in wider use in the company. Otherwise future process improvement efforts were planned in other areas in software engineering.

As was explained above, a common feature in all these companies was that no company seriously considered trial use, assessing different alternatives, or backing off from the steps taken. Instead, all the companies implemented the changes with the mindset that “this is how we will operate in the future.” A central difference between the approaches is, of course, the amount of investments – only a large company can afford the approach taken by Company A. Both the small companies avoided making investments; it seems evident that both of these companies would have had resources for reasonable investments but, as the software architect explained in the context of tool adoption (p. 159), the real issues seemed to be making sure that the investments made – or selected tools – really responded to the companies’ needs. Another observation from the adoption projects is that companies A and B covered all the suggested areas in RE during the project, and planned no further improvements in the RE area in the foreseeable future. Company C with a limited approach, on the other hand, was still hoping to get the full BaRE method in use in the company in the end of the present project. Overall, it seems important that the companies themselves can decide the extent of the improvement effort, the order of implementing different changes, and the schedule for the changes. In the present project the small companies seemed to prefer implementing the changes in small increments as part of daily work while the large company seemed to prefer a project like approach for the changes. Despite the way the changes in infrastructure were made and the training was implemented, all the people seemed to prefer learning by doing rather than by studying. Finally, especially in the small companies it was evident at all times that the daily business had a priority over the process improvement effort; thus any schedules and plans were always just informative and in practice changes were implemented when possible.

The experiences from the adoption processes can be summarized as follows. First of all, all the documents should be available in the adopters’ native language. In the present study the translations and use of English language in the documentation seemed to complicate the adoption considerably. Further, all the method users should participate in high level presentations that provide an overall picture of the method, its background, goals, and principles. The independent study of the method documentation should be supplemented with predefined focused training sessions and packages. In the course of the present study the training sessions were discussed, but in the absence of predefined packages the implementation

seemed too complicated; the interest in custom course development work was generally low and, consequently, no actual courses were developed during the present study. The problems that seemed to halt the adoption projects were not hard to solve for a knowledgeable person but for a first time user they really could prove fatal. Thus easy availability of support through telephone or email as well as regular visits to adopter sites seemed easy ways to assure that the projects were progressing smoothly. This availability of a true expert in the problem domain appeared important especially in the cases where the change agents had not yet proven their technical leadership.

Innovation Decision Process. Reflecting the present study outcomes on the five different stages in an innovation decision process (Rogers 1995, p. 162) (p. 32) it is clear that in the present study the focus was on the knowledge stage. In practice most of the discussions above reflect the knowledge transfer activities in the project. The persuasion stage was practically non-existent since the companies were basically already aware of their development needs; it is suspected that the conducted evaluations, discussions, and general presentations on software engineering in general and requirements engineering in particular increased the awareness and understanding of the companies to the extent that they were ready to move to actions. In the absence of actual decision making criteria in the companies it can be suspected that the decision to focus on RE improvement actions was caused by two facts: first, deficiencies in the RE practices were evident through the initial contacts, and second, the RE practice improvement actions were the first concrete actions offered to the companies that were expected to have direct influence on the company practices. In the implementation stage the companies were provided further information and support but actual implementations were done by the companies themselves. The last stage of confirmation is expected to be continuous in the two small companies following an installment adoption approach. In the large company, however, this stage is expected to be of lower importance since the large company approach based on return on investment calculations requires longer evaluation cycles in order to get the returns recorded for proper evaluation.

Overall, the innovation decision process appeared quite straightforward in the two small companies: all the people involved in the decision making were working in the same office, the roles and competences of the people were clear to everyone, and the decision making in general appeared quick and straightforward. This is also reflected on the stages the small companies seemed to go through in the adoption phase (Figure 6, p. 33). Instead of having the *trial use* stage before adoption, as suggested by Garcia (2002), the small companies seemed rather to have *commitment* and *installation* stages followed by adoption as suggested by Zahran (1998, p. 212). This situation makes theoretical models on decision making appear rigid, and their fit in this environment is questionable. In the large company with multiple levels of decision makers, such models appear more relevant even though in the present study the fit between the model and reality still seems somewhat questionable.

Characteristic Problems in SPI. Chapter 4 closed with a claim that the characteristic problems in small organizations with low maturity are lack of resources and uncertainty about SPI effort payback. In the three conducted case studies all the companies reported a lack of internal slack resources and the two small ones also managed the present study without any financial investments in RE. In the third company, the large one, the situation was clearly different and even large investments could be made with the large company resources. The lack of expertise was most evident in Company B but it is clear that all the participating companies in the present study did utilize external expertise in the improvement efforts. Thus even though the degree

may have differed between the companies, they all lacked expertise to some extent. Finally, the payback uncertainty was evident in the small companies. As mentioned above, the small companies did not make any financial investments, and the discussions indicated that costs need to be carefully considered in these companies. Company C even rejected the possibility use one of the leading configuration and change management system without license costs since they would have needed to set up a Windows-NT server for it. In the large company the payback uncertainty was not such a big issue since there was a standard way to address it with the return on investment calculations. Overall, it can be estimated that these issues were present in the small companies and many of them were also present in the large company. And as claimed above, the BaRE method addressed the majority of the thirteen characteristics developed from these problems. Therefore, it can be claimed that the BaRE method addresses these issues.

8.5. Summary

In this chapter the empirical evaluation of the BaRE method was reported. The evaluation was based on three case studies in three companies; two of the companies were small software houses from which one was living on a software product and the other one on bespoke system development. The third one was a small IT department of a large metal industry corporation. The evaluation project was conducted in two phases and all the companies were interviewed in the beginning of the project, between the phases, and after the completion of the project. In addition to this, other contacts with the companies were kept as needs or opportunities arose.

Based on the company documents and conducted interviews it is evident that changes took place in these companies during the present study. Two of the companies adopted the BaRE method as a whole while the third company adopted only the requirements document template and training material from the method. The two companies with a full-scale adoption scheme adopted, in addition to these two elements, new requirements management tools, new change management processes, and developed requirements with the new template and practices. These two companies also reported at the end of the evaluation project that they did not plan to develop requirements engineering any further but would try to extend the adoption of the new company infrastructure and practices to a wider scale and institutionalize them as real company standards. The third company, on the other hand, reported changes in the way they worked and appeared quite satisfied with their new requirements document template even though it was not yet claimed as a company standard. Despite this, the new document layout was adopted also in other company documents. At the end of the present project, however, the requirements engineering practices in this company had not changed very much in general, and a desire to extend the adoption of the BaRE method was reported.

An analysis for the reasons leading to the changes above supports the claim that they were by and large caused by the adoption of the BaRE method or some of its elements. The study outcomes, on the other hand, were also influenced by environmental and organizational contexts and the technological innovation of the study. In the environmental context the external information sources, or more specifically the author of this thesis, and business partners – the RM tool vendor in the case of Company A – were identified. In the organizational context the management awareness and attitude appeared to be a central factor contributing to the adoption decisions. And finally, from the point of view of technological innovation the key characteristics of the available BaRE method were identified: relative advantage, compatibility, ease of use, trialability, comprehensiveness, readiness-to-use, and adaptability.

A study of the adoption processes in the three companies provided some insight into how these factors work together. First of all, it is noted that the large company chose to make all the changes in a short period of time in a project like manner even though no actual project was established. The small companies, on the other hand, made the changes as part of their daily work and projects. Since the focus of the present study is on small companies, the rest of this discussion approaches the topic from the latter point of view. It seems that changes should be packaged in easy and ready-to-use units that are compatible with the current company equipment and practices since this makes the installment approach (trialability) in adoption possible and efficient. It is evident that the compatibility requirement is very demanding and thus a realistic approach is to strive after compatibility by focusing on a specific domain and allowing for adaptations so that the desired level of compatibility can be achieved as easily as possible. The adaptability, on the other hand, makes it possible to increase the usability of the approach in domains other than the original one. One of the most important things, however, is to keep in mind that the new approach should provide relative advantage to the current approach. The identification of this characteristic can be eased by making the approach a comprehensive one since it makes it possible to get an overall picture of the approach quickly and also makes the study of the perceived problem areas possible in detail.

From the organizational point of view the studied projects were conducted with clearly lower priority than actual production work. Thus, to keep the improvement effort running despite other activities, it is important that the management provides a clear sponsorship for the effort and, preferably, also a senior person in the staff to work as a champion for it. At the employee level, it is equally important that everyone is aware of the development needs and that there is a desire to improve the situation. Finally, the improvement actions are usually done internally by the company people, but an external specialist working as an additional champion and/or as a change agent may prove invaluable for the success of the whole effort.

9. CONCLUSIONS

The present study has approached the question *how to ease the adoption of basic systematic RE practices in small organizations* from different viewpoints. Since a short answer to the question would be too vague to be really useful, only answers to the four detailed research questions are provided in this chapter including the fourth question which has not yet been addressed: *How to develop easy to adopt methods?* Before going into the detailed answers and conclusions of the study, an overview of the whole study is provided.

Chapter 1 introduced the study and Chapter 2 described the terms and definitions used in it. In Chapter 3, findings from literature surveys were summarized from three different viewpoints: software projects and requirements, research areas of interest in this study, and factors affecting method development. Chapter 4 initiated reporting the findings of the study by describing a state-of-the-practice investigation in RE in twelve Finnish companies and, building on the findings of the investigation, identified characteristic problems in small and medium sized companies with low maturity: lack of resources (money, time, and expertise) and uncertainty about process improvement effort payback. In Chapter 5, a solution concept was developed based on the earlier findings outlining thirteen characteristics for an actual solution; these characteristics were classified as ones providing focus on the solution, diffusion of innovation characteristics, and other characteristics that have not received serious attention in previous studies. Chapter 6 introduced the developed solution, the basic RE method BaRE. Since the method has been documented in full elsewhere, this chapter introduced only the most important aspects of the method.

Chapter 7 focused on the literature based evaluation of the BaRE method against four previous studies which included the REAIMS Top Ten guidelines, key issues in RE tool and technique selection, software project success factors, and software project risk factors. These comparisons were supplemented with a study of other documented RE methods providing support for the overall novelty of the solution. As a conclusion from these evaluations the BaRE method was judged to possess the potential to address many central issues in software development and RE in particular. Finally, Chapter 8 provided an account of three case studies where the BaRE method or some of its elements were adopted in industry settings. These case studies provided detailed information about how small organizations make adoption decisions on technological innovations, what factors affected those decisions in the present study, and what actually happened in the course of the empirical evaluation. The following sections provide more detailed discussion on the findings of this study. Section 9.1 approaches the findings from the point of view of the research questions and closes with the limitations of the study, Section 9.2 discusses the generalizability of the results, and Section 9.3 offers suggestions for future research.

9.1. BaRE – an Easy to Adopt Method for Requirements Engineering

In this section, answers to the first three research questions are provided and the limitations of the conducted study are reported. The first research question was the following:

What are the desirable characteristics of a requirements engineering method that would make it easy for small organizations to adopt it?

This question was first approached in Chapter 4 by the conducted state of the practice investigation and the literature study. The overall outcome was that small organizations with low maturity have two characteristic problems: lack of resources (i.e., money, time, and expertise) and uncertainty about process improvement effort payback (Section 4.2, p. 79). The solution concept formation developed altogether thirteen characteristics that seemed important for a potential solution to the first research question (Table 17, p. 96). Finally, the empirical evaluation of the method concluded that there were seven characteristics that had a direct affect on the present study outcomes and helped to change the RE practices in the participating companies: relative advantage, compatibility, ease of use, trialability, comprehensiveness, readiness-to-use, and adaptability (Table 43, p. 173).

The second research question was the following:

Is it possible to develop an easy to adopt method for requirements engineering?

Chapter 6 summarized the developed method and a more detailed description of the method can be found elsewhere (Nikula 2002b). Thus the method construction aspect of this question is easily verified and the focus can be moved to the ease of its adoption which was studied in three separate organizations. The actual adoption of the method, or its elements, was both questioned directly from the participants and verified through changes in the RE infrastructure and working practices in the companies.

Two of the participating companies adopted the BaRE method in full while one company developed a new requirements document template based on the method and adopted the training material from the method (Table 32, p. 135). The two companies with a full-scale adoption scheme adopted a new requirements management tool from other sources and developed their own change management practices. In the course of the study, these two companies stated that the BaRE method was their standard way of doing requirements engineering with some documented company specific adaptations. Thus two of the companies in the present study established a comprehensive company specific RE infrastructure during the evaluation project (Table 36, p. 139). The third company developed their own adaptation of the suggested requirements document and adopted it in daily use in the company (Table 37, p. 140). Otherwise the changes in working practices were not extensive, which is at least partly due to the fact that, for example, many of the suggested techniques were already used in the companies (Table 40, p. 165). The most important changes in the working practices were reported by the system analyst in Company A (p. 138) who claimed that her approach to requirements development had changed to a more systematic one, and by the project manager C3 in Company C (p. 140) who claimed that the co-operation had gotten tighter and more effective in their company. Overall, the goals and the main achievements in the companies were summarized in Table 38 (p. 144). In the end of the study, the companies that implemented the BaRE method in full reported that they did not plan any more changes in the RE area, but only planned to extend the use of the method, while Company C with only a partial adoption scheme was hoping to adopt new elements from the BaRE method in the company use.

In addition to the fact that the companies adopted the BaRE method or some of its elements, they also made a number of changes to the adopted elements. The changes are the most evident in the way the companies chose to use tools to support their RE. Company A adopted a commercial requirements management tool and adapted it to their new working practices; they also specified a new change management process based on this and another tool. Company B

adopted a freeware requirements management tool and started managing their changes with it; later on during the project also the requirements for a new software version were collected with this tool. However, the actual requirements document was based on the text processor document template provided with the BaRE method. In Company C the requirements document template was refined based on the BaRE template, but another template was also developed to support another text processor system.

Despite the completed changes the companies used fairly little time in the RE development work. The small companies B and C used less than two person weeks and the large company A used less than 2.5 person months for the RE development work (Table 39, p. 145). The calendar times for these changes were also fairly short. Company A established its RE infrastructure in only six months from the beginning of the project; Company B got its changes under control in four months and established the whole RE infrastructure in ten months; and Company C developed a company specific RD template in nine months but completed projects with an earlier version of the template in six months.

Considering the scope of the changes in the companies, the efforts and calendar time used are not very extensive. Thus, based on the available information about the BaRE method adoption in these companies, it can be claimed that the method was fairly easy to adopt.

The third research question was the following:

What are the desirable characteristics of the introduction process of an easy to adopt requirements engineering method in small organizations?

Analyzing the conducted case studies there are a number of matters that appear important from the point of view of the adoption process. First of all, it became evident that in the small companies the daily business had higher priority than process improvement actions. As a consequence, the changes were implemented aside other daily tasks as small increments or installments which were selected based on short term needs. In the same vein, the production tasks dictated the amount of time available for the improvement efforts and their implementation schedule. Finally, this also meant that the changes were taken into production use directly after or during their implementation. Overall, the small companies were reluctant to make investments, which was on one occasion explained by the uncertainty of how well an investment would pay itself back (cf. constraints on compatibility, p. 159).

Company A differed from the small companies significantly in the general approach to changes. It implemented all the infrastructure related aspects of the effort in one step in a short period of time. A fairly long study and planning period preceded the actual investment decision, but thereafter the changes were implemented quickly in the whole company and all the people participating in the RE activities were provided related basic training. The actual amount of investments was not considered so essential but a more important aspect was the return on investments that was estimated with calculations. On the working practices side the company did not get as far as the two small companies and only the change agent got proper exposure to the new infrastructure.

Considering the common preferences in all the companies, two observations can be made. First, all the companies implemented the process improvement actions based on their own needs,

resources, and schedules even though they tried to comply with the BaRE evaluation project goals and plans. Second, all the participating people seemed to prefer learning by actually working on real production projects rather than by just studying.

Even though the discussion so far has focused on the characteristics of the BaRE method and the adoption process, these were not the only factors affecting the adoption decisions and the outcomes of the present study. In the organizational context the management awareness and attitude was identified as a clear contributing factor to the project outcomes. In the small companies the managing directors clearly had the role of the project sponsor to assure that it had the official company backing. In companies A and C it was also easy to identify the project champions while companies A and B had change agents named clearly. Based on the overall Company A behavior, it is evident that there was a project sponsor somewhere even though this was never verified; the lack of a clear change agent in Company C, on the other hand, could in part explain the limited scope of the effort in the company. In Company B no champion was identified but there were two people that could have served for or shared this role – the managing director of the company or the author of this thesis. Especially the change of the whole software development personnel halfway through the project created a need for a champion outside the development team. The senior software engineer provided support for the view that the author of this thesis could have served in this role (p. 150). The role of the author of this thesis was also otherwise found important for the conducted study (cf. system analyst comment on p. 150). The collected data provides further evidence on the importance of the tool consultant in the case of Company A (p. 172). Thus, in addition to the technical aspects of the innovation, it seems that both the management and the availability of external expert support played important roles in the outcomes of the present study.

As the single most striking observation from the empirical evaluation it is noted that interest in making comparisons, evaluations, or assessments between different solution alternatives was very limited. In all the companies it was always evident that all the actions were done to achieve the next phase in the decision making concerning the process improvement endeavor. Before the decision making different levels of background study and information gathering took place, but only in the large company efforts were made to figure out the most suitable solution from competing ones. Overall, the decisions were made quickly based on the available information and previous experiences.

Limitations of the study. The conducted study has numerous limitations. First of all the conducted study was an exploratory one where the detailed goals, design, and implementation were intertwined with each other. Thus the conducted study lacked the clear a priori goal setting possible with a confirmative study. The fact that a researcher, the author of this thesis, participated in the case studies has clearly affected the study outcome. Since this situation could not be avoided, the extent of the participation was controlled and support was provided as a response to taken actions based on the developed material common to all participants. Consequently, all the participating companies had the same material in use and, on the other hand, the company contacts were scheduled and conducted in a similar manner (Section 1.3.2, p. 20). Further, the role of the researcher in the study outcome was questioned directly by the participating companies (Section 8.3.2, p. 149); it is clear that the answers to this question may have been biased since they were presented by the researcher himself, but due to the small scale and qualitative nature of the study there were no real alternatives to this approach. The time period of the conducted case studies, eleven months, is a fairly short one to implement and identify persistent changes in any company. The different characteristics of the participating

organizations and the different application types they developed complicate the assessment of the different factors in the study outcome. When these properties are combined with the small number of companies, it is clear that the results of this study are much more suited as initial evidence for larger scale studies to gain more insight about the study findings. The developed BaRE method is clearly not an ideal easy to adopt method but has many shortcomings. Finally, most of the identified characteristics for the method are subjective and their measurement is not straightforward especially in a qualitative study.

9.2. Guidelines for Easy to Adopt Method Development

The BaRE method was developed to show on a practical level how to do RE in a systematic way and document the key issues in software development in a lightweight manner. On the basis of the practical experiences, it appears justified to develop methods for other software engineering process areas following the same principles. The actual development work of an easy to adopt method should pay close attention not only to the method characteristics and components but also to the adoption process. Thus the rest of this section provides an answer to the fourth research question of the present study:

How to develop easy to adopt methods?

Section 5.2 (p. 90-) introduced thirteen characteristics for an easy to adopt method but only seven of them were found to have had direct affect on the outcomes of the present study. However, this does not make the other six characteristics obsolete by and large – rather their impact in the conducted study was estimated so limited that there are no grounds for claiming the opposite. Thus, with the available information, it appears justified to suggest that an easy to adopt method development should specify requirements describing its domain of applicability, innovation characteristics, and other characteristics (cf. Table 44).

Table 44. Characteristics of an introduction level easy to adopt method

Application Domain	Innovation Characteristics	Other Characteristics
3-Tier Architecture Focus	Relative Advantage	Comprehensiveness
Small Application Focus	Compatibility	Readiness-to-Use
Small Team Focus	Ease of use	Adaptability
Introduction Level Focus	Trialability	
	Result Demonstrability	
	Visibility	

The specificity of the application domain is important as it makes it possible to base the method development on the domain characteristics. The innovation characteristics have been found in earlier studies to play an important role in adoption decisions, and finally the group *other characteristics* describes novel characteristics in the present study. These characteristics were found important in the present study in which the goal was to make the adoption of a basic level RE method in an industrial setting easy. Thus there is reason to believe that a comparable introduction level method could benefit from these characteristics also; in the case the level of

maturity – or more generally the application domain – changes considerably, the relevance of these characteristics may become questionable.

Since the BaRE method was developed to address RE specific issues, components for a more general method should be generalized to templates, practices, tool support, and training. The templates component refers to the document and other possible templates that serve as the basis for producing project deliverables. Thus templates are often the most concrete part of a method, and suggesting templates that can be taken in use quickly seems to provide the most tangible benefit to the daily work. The term *practices* refers to activities that utilize processes, techniques, and checklists to develop the actual data shown in a document, and thus they form the basis for a systematic way of working that actually produces results. To serve inexperienced users the most, the practices should be simple, possibly focus on naming techniques that are already used, and suggest new simple and effective techniques to supplement current ones. This approach starts to develop a structure to working practices and makes continuous improvement a natural way of working. Efficient processes, however, tend to require numerous routine tasks that make their completion tedious unless they are supported by tools. Thus tools are used to automate routine tasks and ease following processes so that they become standard ways to operate in a company. The selection criteria for tools are often very organization dependent, and therefore a method should introduce the spectrum of available tools and suggest differentiating factors between them. Providing a tool that is integrated with the method is, of course, an advantage, but in the case of the BaRE method the lack of an integrated tool did not prove a threshold issue for the method adoption. The last component, training, should be understood in broad terms covering documentation, presentations, and specialist support. These mechanisms should be used to provide introduction and background information, practical advice on the work, and troubleshooting support. The documentation is often the most important part for the actual learning, but providing a context for the method and resolving issues before they become problems is crucial for the efficient adoption of any method.

The adoption process is not a part of a method but still intimately related with it. Thus, considering the experiences from the BaRE method adoption processes, an ideal adoption process can be outlined in retrospect as follows. First of all, the adopting organization needs to be aware of the existing problems both on the management and employee levels, and a consensus should exist on the need to address the problems in the company. Once an easy to adopt method addressing the central problems is available, the match between the organizational needs and available technology should be assessed. If the problem and the solution are found to provide an acceptable match, practical adoption actions can be initiated. All the people that get involved in the process improvement effort should participate in high level presentations of the effort so that they are aware of the background, goals, and principles of the process improvement effort and the method. The people who are to use the new method should study the method documentation independently for a day or two. This self-study should be supported by predefined training sessions and packages that focus on detailed areas of the method. The actual adaptation and use of the method in practice should be then conducted.

Based on the experiences from the BaRE evaluation project external expertise should be available for an adoption project. In the conducted project the following approach seemed to work well: an expert suggested how the actual adoption could happen, and the adopting companies adapted this proposal to their needs. The actual implementation responsibility was best left to the adopting companies, but close support and easy access to an external specialist appeared critical in the conducted project, and therefore, it seemed that even some of the

responsibility of the whole effort was born by the external specialist. The actual amount of interaction between the specialist and the companies was limited, and a much more critical factor seemed to be easy access to the specialist.

Overall, the original BaRE method adoption strategy is worth considering for any company planning process improvement efforts – the method adoption should be started at the project level (Section 6.2.5, p. 111). If the method proves useful, it can be expected to be adopted on a wider scale in other projects, which might even lead to the adoption of the method as a company wide standard. This approach may appear inefficient from the organizational point of view, but if it results in real improvements, as happened in the BaRE evaluation project, the achieved improvements can outweigh the seemingly slow process.

9.3. Future Research

Since the present study addresses topics that can be considered interesting both from the academic and industrial points of view, it seems justified to extend and deepen the handling of these topics in the future. Two general questions of central interest from the point of view of the present study are the following:

1. What is the state-of-the-practice in software development? Since a pre-emptive answer to this question is clearly not a realistic goal, conducting a regionally representative study could provide further insights about the relative importance of different software engineering processes and different approaches to software engineering as well as perceived problems in software engineering and process improvement in general.
2. What are the central issues in requirements engineering and software process improvement? In the present study these issues were addressed mostly based on literature sources, so, conducting in-depth studies in these areas could help in revising the suggested method and its introduction process to better fit the actual problems.

Considering the BaRE method there are some clear future research needs. Namely, the conducted explanatory study provides a good basis for establishing a project of a confirmative nature. Thus it would make sense to conduct a larger scale BaRE adoption project were a larger number of companies would be supported in the BaRE method adoption. Before this kind of project can be conducted, however, the current experiences should be reflected on the method and its adoption process. In particular this means the following:

1. Revise the BaRE method to increase its ease of adoption. The two most important aspects in this respect are the availability of tool support for requirements management and the development of standard training courses to support the method learning. Further ideas for the method development are documented in Appendix 8. (Change Requests for the BaRE version 1.0).
2. Revise the BaRE method introduction process. The introduction process should be revised to reflect the experiences gained in the present study and, on the other hand, the method adoption process should be supplemented with appropriate evaluation mechanisms to be able to track the progress, decisions, and factors affecting the decisions.

Finally, the feasibility of extending the easy to adopt nature of the BaRE method in other environments should be studied in more detail. Two fundamental questions in this area need further study:

1. Is the ease of adoption extendable to other domains? This question is related in particular to the introduced constraints on the BaRE method (3-tier architecture, small application, small team, and introduction level focus) and to the process area (requirements engineering vs. project management or testing etc.).
2. How should an easy to adopt method covering the whole software development phase be constructed? Addressing individual process areas is basically a question of replicating and adapting the suggested method to another process area, but addressing the whole software development phase presents another problem. Namely, can the suggested concept be extended as such to the whole software development phase or does the ease of adoption make it necessary to develop even less lightweight approaches in different areas to avoid making the overall approach too complex?

The future research needs can also be viewed from the point of view of the example ready-to-use and easy to adopt artifact described in Section 2.4 (p. 32) – the mobile phone. From the practical point of view, the BaRE method is not as easy to adopt as a regular mobile phone but the question is how easy to adopt should it be? In the case of a mobile phone, calls can be made and received within minutes from opening the package; in the case of the BaRE method requirements documentation development can also be started within minutes after getting hold of the method documentation and templates. Achieving the full benefits of the advanced features of a mobile phone like the calendar, camera, multimedia messages, and data transfer features requires both adaptation of the basic phone to the user environment (e.g. connection service provider, geographic location, etc.) and learning from the user. In the same vein, conducting state-of-the-art requirements engineering requires careful selection of automated tools, adaptation of the method to the specific requirements in the application and organizational contexts etc. as well as learning from the user. Reflecting on the evolution of the mobile phones and their ease of adoption, it seems evident that increasing the ease of adoption of software engineering methods is a never ending journey and it is high time to start working towards this goal.

EPILOGUE

This thesis started with a story about RE practice and research. Having worked in this area for quite a while now, it is time to revise the end of the original story as follows:

“Okay. I’ve convinced the people back home that our problem is poor attention to requirements, and that the solution is to make Requirements Engineering a top priority. I have just one question for you: What do we do now?”

A long silence followed. The researcher stared into his teacup thoughtfully.

“Hmm... Why don’t you try the BaRE method, it gives you a head start in doing RE.”

REFERENCES

- AgileAlliance (2002). Agile Alliance Home Page. www.agilealliance.com. Accessed July 18, 2002.
- Ahituv, N., M. Zviran, and C. Glezer (1999). "Top Management Toolbox for Managing Corporate IT." Communications of the ACM **42**(4): 93-99.
- Ajzen, I. (1991). "The Theory of Planned Behavior." Organizational Behavior and Human Decision Processes **50**(2): 179-211.
- Alford, M. (2002). Software Requirements Engineering Methodology. Encyclopedia of Software Engineering. J. J. Marciniak Ed. New York, John Wiley & Sons. Vol 2, pp. 1630-1642.
- Alford, M. W. (1978). "Software Requirements Engineering Methodology (SREM) at the Age of Two." Computer Software and Applications Conference, IEEE Computer Society. pp. 332-339.
- Andersen, N. E., F. Kensing, J. Lundin, L. Mathiassen, A. Munk-Madsen, M. Rasbech, and P. Sørsgaard (1990). Professional Systems Development: Experience, Ideas and Action. New York, Prentice Hall.
- ANSI/IEEE Standard 830 (1998). ANSI/IEEE Standard 830-1998. IEEE Recommended Practice for Software Requirements Specifications. The Institute of Electrical and Electronics Engineers Ed. New York, NY, IEEE Computer Society Press.
- ANSI/IEEE Standard 1233 (1998). ANSI/IEEE Standard 1233-1998. IEEE Guide for Developing System Requirements Specifications. The Institute of Electrical and Electronics Engineers Ed. New York, NY, IEEE Computer Society Press.
- ANSI/IEEE Standard 1362 (1998). ANSI/IEEE Standard 1362-1998. IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document. The Institute of Electrical and Electronics Engineers Ed. New York, NY, IEEE Computer Society Press.
- Antón, A. I. (1996). "Goal-Based Requirements Analysis." Second IEEE International Conference on Requirements Engineering, Colorado Springs, Colorado, IEEE Computer Society. pp. 136-144.
- Antón, A. I., R. A. Carter, A. Dagnino, J. H. Dempster, and D. F. Siege (2001). "Deriving Goals from a Use-Case Based Requirements Specification." Requirements Engineering **6**(1): 63-73.
- Association for Computing Machinery (1998). The ACM Computing Classification System (1998, Valid in 2003): J.1 ADMINISTRATIVE DATA PROCESSING. <http://www.acm.org/class/1998/J.1.html>. Accessed June 25, 2003.
- Attewell, P. (1992). "Technology Diffusion and Organizational Learning: The Case of Business Computing." Organizational Science **3**(1): 1-19.
- Avison, D. and G. Fitzgerald (2003). Information Systems Development: Methodologies, Techniques, and Tools. 3rd edition. Berkshire, United Kingdom, McGraw-Hill Education.
- Baddoo, N. and T. Hall (2002). "Motivators of Software Process Improvement: An Analysis of Practitioners' Views." Journal of Systems and Software **62**(2): 85-96.
- Baddoo, N. and T. Hall (2003). "De-Motivators for Software Process Improvement: An Analysis of Practitioners' Views." Journal of Systems and Software **66**(1): 23-33.

- Basili, V. (1989). "Software Development: A Paradigm for the Future." 13th Annual International Computer Software and Applications Conference, IEEE Computer Society. pp. 471-485.
- Baskerville, R. L., B. Ramesh, L. Levine, J. Pries-Heje, and S. Slaughter (2003). "Is Internet-Speed Software Development Different." IEEE Software **20**(6): 70-77.
- Bean, A. S., R. D. Neal, M. Radnor, and D. A. Tansik (1975). Structural and Behavioral Correlates of Implementation in U.S. Business Organizations. Implementing Operations Research/Management Science. R. L. Schultz and D. P. Slevin Eds. New York, American Elsevier.
- Beck, K. (1999a). "Embracing Change with Extreme Programming." Computer **32**(10): 70-77.
- Beck, K. (1999b). Extreme Programming Explained: Embrace Change. Boston, Addison-Wesley.
- Beecham, S., T. Hall, and A. Rainer (2003). "Software Process Improvement Problems in Twelve Software Companies: an Empirical Analysis." Empirical Software Engineering **8**(1): 7-42.
- Berry, D. M., K. Daudjee, J. Dong, I. Fainchtein, M. A. Nelson, T. Nelson, and L. Ou (2004). "User's Manual as a Requirements Specification: Case Studies." Requirements Engineering **9**(1): 67-82.
- Berry, D. M., E. Kamsties, and M. M. Krieger (2003). From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. A Handbook Version 1.0. <http://se.uwaterloo.ca/~dberry/#Handbook>. Accessed September 17, 2004.
- Beyer, H. and K. Holtzblatt (1998). Contextual Design: Defining Customer-Centered Systems. San Francisco, CA, Morgan Kaufmann.
- Blanchard, B. S. and W. J. Fabrycky (1990). Systems Engineering and Analysis. 2nd edition. Englewood Cliffs, NJ, Prentice Hall.
- Blili, S. and L. Raymond (1993). "Information Technology: Threats and Opportunities for Small and Medium-Sized Enterprises." International Journal of Information Management **13**(6): 439-448.
- Boehm, B. (1984). "Verifying and Validating Software Requirements and Design Specifications." IEEE Software **1**(1): 75-88.
- Boehm, B. (1988). "A Spiral Model of Software Development and Enhancement." Computer **21**(5): 61-72.
- Boehm, B. (2002). "Get Ready for Agile Methods, with Care." Computer **35**(1): 64-69.
- Braun, C. L. (2002). Reuse. Encyclopedia of Software Engineering. J. J. Marciniak Ed. New York, John Wiley & Sons. Vol 2, pp. 1197-1214.
- Breitling, H., P. Becker-Pechau, and S. Rook (2002). "Tutorial Notes T03: Agile Requirements Engineering." IEEE Joint International Conference on Requirements Engineering, Essen, Germany, IEEE Computer Society.
- Brinkkemper, S. (1996). "Method Engineering: Engineering of Information Systems Development Methods and Tools." Information and Software Technology **38**(4): 275-280.
- Brooks, F. P., Jr (1987). "No Silver Bullet - Essence and Accidents of Software Engineering." Computer **20**(4): 10-19.
- Brooks, F. P., Jr (1995). The Mythical Man-Month. Anniversary Edition. Reading, Massachusetts, Addison-Wesley.

- Bustard, D. W. and F. G. Wilkie (1999). "Soft Systems and Use-case Modeling: Mutually Supportive or Mutually Exclusive." 32nd Annual Hawaii International Conference on Systems Sciences, Hawaii, IEEE Computer Society. pp. 1-8.
- Calvo-Manzano Villalón, J. A., J. C. Bravo, and T. San Feliu Gilabert (1997). "Software Process Improvement: MESOPYME Model and Method." Journal of Computing and Information Technology **5**(3): 159-165.
- Calvo-Manzano Villalón, J. A., G. Cuevas Agustín, T. San Feliu Gilabert, A. De Amescua Seco, L. García Sánchez, and M. Pérez Cota (2002). "Experiences in the Application of Software Process Improvement in SMES." Software Quality Journal **10**(3): 261-273.
- Caron, J. R., S. L. Jarvenpaa, and D. B. Stoddard (1994). "Business Reengineering at CIGNA Corporation: Experiences and Lessons Learned from the First Five Years." MIS Quarterly **18**(3): 233-250.
- Carter, R., J. Martin, B. Mayblin, and M. Munday (1988). Systems, Management and Change: A Graphic Guide. London, Paul Chapman Publishing.
- Charette, R. (2001). "The Decision is in: Agile Versus Heavy Methodologies." e-Project Management Advisory Service: Executive Update **2**(19).
<http://www.cutter.com/research/freestuff/apmupdate.pdf>.
- Chatzoglou, P. D. (1997a). "Factors Affecting Completion of the Requirements Capture Stage of Projects with Different Characteristics." Information and Software Technology **39**(9): 627-640.
- Chatzoglou, P. D. (1997b). "Use of Methodologies: An Empirical Analysis of Their Impact on the Economics of the Development Process." European Journal of Information Systems **6**(4): 256-270.
- Chau, P. Y. K. and K. Y. Tam (1997). "Factors Affecting the Adoption of Open Systems: An Exploratory Study." MIS Quarterly **21**(1): 1-24.
- Checkland, P. (1981). Systems Thinking, Systems Practice. Chichester, John Wiley & Sons.
- Checkland, P. and J. Scholes (1990). Soft Systems Methodology in Action. Chichester, John Wiley & Sons.
- Cheesman, J. and J. Daniels (2000). UML Components: A Simple Process for Specifying Component-Based Software. Boston, Addison-Wesley.
- Cockburn, A. (1997). "Goals and Use Cases." Journal of Object-Oriented Programming **10**(7): 35-40.
- Cockburn, A. (2002). Agile Software Development. Boston, Addison-Wesley.
- Constantine, L. (2001). "Methodological Agility." Software Development **21**(6): 67-69.
www.sdmagazine.com.
- Cronbach, L. J. (1971). Test Validation. Educational Measurement, R. L. Thorndike Ed. Washington, D.C, American Council on Education. pp. 443-507.
- Curtis, B., W. E. Hefley, and S. A. Miller (2001). People Capability Maturity Model (P-CMM). Maturity Model CMU/SEI-2001-MM-01. Pittsburgh, PA, USA, Software Engineering Institute: 735 p. Available at
<http://www.sei.cmu.edu/publications/documents/01.reports/01mm001.html>.
- Curtis, B., W. E. Hefley, S. A. Miller, and M. Konrad (1997). "Developing Organizational Competence." Computer **30**(3): 122-124.

- Curtis, B., H. Krasner, and N. Iscoe (1988). "A Field Study of the Software Design Process for Large Systems." Communications of the ACM **31**(11): 1268-1287.
- Cusumano, M. A. and R. W. Selby (1996). Microsoft Secrets. London, HarperCollinsPublishers.
- Cutter Consortium (2001a). Big IT Projects Meet Schedules and Budgets as Well or Better Than Small Ones, Cutter Consortium. www.cutter.com. Accessed January 31, 2002.
- Cutter Consortium (2001b). Research Findings, Cutter Consortium. www.cutter.com. Accessed July 18, 2002.
- Czarnecki, K. (2002). Domain Engineering. Encyclopedia of Software Engineering. J. J. Marciniak Ed. New York, John Wiley & Sons. Vol 1, pp. 433-444.
- Damian, D. E. H., A. Eberlein, M. L. G. Shaw, and B. R. Gaines (2000). "Using Different Communication Media in Requirements Negotiation." IEEE Software **17**(3): 28-36.
- Davis, A. M. (1993). Software Requirements: Objects, States, and Functions. Prentice Hall.
- Davis, A. M. (1995). 201 Principles of Software Development. New York, McGraw-Hill.
- Davis, A. M. (2002a). Alan Mark Davis, Requirements Bibliography. <http://www.uccs.edu/~adavis/>. Accessed July 17, 2002.
- Davis, A. M. (2002b). "Tutorial Notes T01: Just Enough Requirements Management (JERM)." IEEE Joint International Conference on Requirements Engineering, Essen, Germany, IEEE Computer Society.
- Davis, A. M. and A. M. Hickey (2002). "Viewpoint: Requirements Researchers: Do We Practice What We Preach?" Requirements Engineering **7**(2): 107-111.
- Davis, F. D. (1989). "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology." MIS Quarterly **13**(3): 318-340.
- Davis, F. D., R. P. Bagozzi, and P. R. Warshaw (1989). "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models." Management Science **35**(8): 982-1003.
- Dearden, A. and S. Howard (1998). "Capturing User Requirements and Priorities for Innovative Interactive Systems." Australasian Computer Human Interaction Conference, IEE Computer Society. pp. 160-167.
- Doherty, N. and M. King (1998). "The Consideration of Organizational Issues During the System Development Process: An Empirical Analysis." Behavior & Information Technology **17**(1): 41-51.
- Dorfman, M. and R. H. Thayer (1990). Standards, Guidelines, and Examples on System and Software Requirements Engineering. Los Alamitos, CA, IEEE Computer Society Press.
- Downward, B. G. (1994). "A Brave New World: Melding Systems and Software Engineering." Fourth Annual International Symposium: Systems Engineering: A Competitive Edge in a Changing World, San Jose, California, USA, International Council on Systems Engineering.
- Dreyfus, H. L. (1991). Being-in-the-World: A Commentary on Heideggers' Being and Time, Division I. Cambridge, The MIT Press.
- DSDM (2002). Dynamic Systems Developmetn Method Home Page. www.dsdm.org. Accessed July 18, 2002.

- Durán, A., A. Ruiz-Cortés, R. Corchuelo, and M. Toro (2002). "Supporting Requirements Verification Using XSLT." IEEE Joint International Conference on Requirements Engineering, Essen, Germany, IEEE Computer Society. pp. 165-172.
- Dybå, T. (2000). "An Instrument for Measuring the Key Factors of Success in Software Process Improvement." Empirical Software Engineering **5**(4): 357-390.
- Dybå, T. (2002). "Enabling Software Process Improvement: An Investigation of the Importance of Organizational Issues." Empirical Software Engineering **7**(4): 387-390.
- Easterbrook, S. and M. Chechik (2001). "A Framework for Multi-Valued Reasoning over Inconsistent Viewpoints." 23rd International Conference on Software Engineering, IEEE Computer Society. pp. 411-420.
- El Emam, K., J.-N. Drouin, and W. Melo, Eds. (1998). SPICE: The Theory and Practice of Software Process Improvement and Capability Determination. Los Alamitos, California, IEEE Computer Society.
- El Emam, K. and N. H. Madhavji (1995). "A Field Study of Requirements Engineering Practices in Information Systems Development." Second IEEE International Symposium on Requirements Engineering, York, England, IEEE Computer Society. pp. 68-80.
- Ewusi-Mensah, K. (1997). "Critical Issues in Abandoned Information Systems Development Projects." Communications of the ACM **40**(9): 74-80.
- Ewusi-Mensah, K. and Z. H. Przasnyski (1991). "On Information Systems Project Abandonment: An Exploratory Study of Organizational Practices." MIS Quarterly **15**(1): 67-86.
- Fayad, M. E., M. Laitinen, and R. P. Ward (2000). "Software Engineering in the Small." Communications of the ACM **43**(3): 115-1118.
- Federal Information Processing Standards (1976). Functional Requirements Document, Data Requirements Document, and System/Subsystem Specification. Guidelines For Documentation of Computer Programs and Automated Data Systems, FIPS PUB 38. N. B. o. S. U.S. Department of Commerce Ed., U.S. Department of Commerce. pp. 195-209.
- Fenton, N., S. L. Pfleeger, and R. L. Glass (1994). "Science and Substance: A Challenge to Software Engineers." IEEE Software **11**(4): 86-95.
- Field, T. (1997). "When BAD Things Happen to GOOD Projects." CIO **15**(18): 55-62. <http://www.cio.com/archive/101597/bad.html>.
- Finnish Standards Association SFS (2000). Layout Of the Document Text Area. Office Documents. Standards. Finnish Standards Association SFS Ed. Helsinki, Finnish Standards Association, SFS.
- Fitzgerald, B. (1998). "An Empricial Investigation into the Adoption of Systems Development Methodologies." Information and Management **34**(6): 317-328.
- Flynn, D. (1998). Information Systems Requirements: Determination and Analysis. 2nd edition. London, McGraw-Hill.
- Forsberg, K. and H. Mooz (1997). System Engineering Overview. Software Requirements Engineering. R. H. Thayer and M. Dorfman Eds, IEEE Computer Society Press. pp. 44-72.
- Fowler, P., M. Patrick, A. Carleton, and B. Merrin (1998). "Transition Packages: An Experiment in Expediting the Introduction of Requirements Management." Third IEEE International Conference on Requirements Engineering, Colorado Springs, Colorado, IEEE Computer Society. pp. 138-145.

- Fuggetta, A. (2000). "Software Process: A Roadmap." Future of the Software Engineering, Limerick, Ireland, ACM Press. pp. 25-34.
- Galliers, R. D. (1992). Choosing Information Systems Research Approaches. Information Systems Research. R. Galliers Ed. Oxford, Blackwell Scientific Publications. pp. 144-162.
- Garcia, S. (2002). "Are You Prepared for CMMI?" Crosstalk: The Journal of Defense Software Engineering(3). <http://www.stsc.hill.af.mil/crosstalk/2002/03/garcia.html>.
- Gause, D. C. (2000). "User DRIVEN Design—The Luxury that has Become a Necessity, A Workshop in Full Life-Cycle Requirements Management." International Conference on Requirements Engineering, Tutorial T7, Schaumburg, IL, USA.
- Gause, D. C. and G. M. Weinberg (1989). Exploring Requirements: Quality BEFORE Design. New York, Dorset House Publishing.
- Gause, D. C. and G. M. Weinberg (1990). Are Your Lights On? How to Figure Out What the Problem REALLY Is. New York, Dorset House Publishing.
- Ginzberg, M. J. (1981a). "Early Diagnosis of MIS Implementation Failure: Promising Results and Unanswered Questions." Management Science **27**(4): 459-478.
- Ginzberg, M. J. (1981b). "Key Recurrent Issues in the MIS Implementation Process." MIS Quarterly **5**(2): 47-59.
- Glass, R. L. and I. Vessey (1992). "Toward a Taxonomy of Software Application Domains: History." The Journal of Systems and Software **17**(2): 189-200.
- Glass, R. L. and I. Vessey (1995). "Contemporary Application-Domain Taxonomies." IEEE Software **12**(4): 63-76.
- Glass, R. L. and I. Vessey (1998). "Focusing on the Application Domain: Everyone Agrees It's Vital, But Who's Doing Anything About It?" Thirty-First Hawaii International Conference on System Sciences, 3, Hawaii, IEEE Computer Society. pp. 187-196.
- Goguen, J. A. and C. Linde (1993). "Techniques for Requirements Elicitation." IEEE International Symposium on Requirements Engineering. pp. 152-164.
- Graham, I. (1998). Requirements Engineering and Rapid Development: An Object-Oriented Approach. Harlow, England, Addison-Wesley.
- Green, G. C. and A. R. Hevner (1999). Perceived Control of Software Developers and Its Impact on the Successful Diffusion of Information Technology. Special Report CMU/SEI-98-SR-013. Pittsburgh, PA, USA, Software Engineering Institute: 118 p. Available at <ftp://ftp.sei.cmu.edu/pub/documents/98.reports/pdf/98sr013.pdf>.
- Green, G. C. and A. R. Hevner (2000). "The Successful Diffusion of Innovations: Guidance for Software Development Organizations." IEEE Software **17**(6): 96-103.
- Greenspan, S. J. (1998). "Delivering Requirements Engineering: Introduction to the Mini-Tutorial." Third IEEE International Conference on Requirements Engineering, Colorado Springs, Colorado, IEEE Computer Society. pp. 128-129.
- Greenspan, S. J. (2001). "Extreme RE: What If There Is No Time for Requirements Engineering." Fifth IEEE International Symposium on Requirements Engineering, Toronto, Ontario, Canada, IEEE Computer Society Press. pp. 282-284.
- Guinan, P. J., J. G. Coopridge, and S. Faraj (1998). "Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach." Information Systems Research **9**(2): 101-125.

- Gupta, D. and N. Prakash (2001). "Engineering Methods from Method Requirements Specifications." Requirements Engineering **6**(3): 135-160.
- Haag, S., M. K. Raja, and L. L. Schkade (1996). "Quality Function Deployment Usage in Software Development." Communications of the ACM **39**(1): 41-49.
- Haikala, I. and J. Märijärvi (1997). Ohjelmistotuotanto. 4. painos (In Finnish). Espoo, Suomen Atk-kustannus Oy.
- Hall, G., J. Rosenthal, and J. Wade (1993). "How to Make Reengineering Really Work." Harvard Business Review **71**(6): 119-131.
- Hall, T., A. Rainer, and N. Baddoo (2002). "Implementing Software Process Improvement: An Empirical Study." Software Process Improvement and Practice **7**(1): 3-15.
- Hammer, M. and J. Champy (1993). Reengineering the Corporation: A Manifesto for Business Revolution. New York, HarperCollinsPublishers.
- Harbaugh, S. (1993). "Experiences in Training Software Engineers to Perform Systems Engineering." Third Annual International Symposium: Systems Engineering in the Work Place, Arlington, Virginia, USA, International Council on Systems Engineering. pp. 783-786.
- Hardgrave, B. C., F. D. Davis, and C. K. Riemenschneider (2003). "Investigating Determinants of Software Developers' Intentions to Follow Methodologies." Journal of Management Information Systems **20**(1): 123-151.
- Hardiman, S. J. (2002). "REDEST - 14 Best Practice SME Experiments with Innovative Requirements Gathering Techniques." IEEE Joint International Requirements Engineering Conference, Industrial Presentations, Essen, Germany, IEEE Computer Society. pp. 53-64.
- Hardy, C. J., J. B. Thompson, and H. M. Edwards (1995). "The Use, Limitations and Customization of Structured Systems Development Methods in the United Kingdom." Information and Software Technology **37**(9): 467-477.
- Harel, D. (1997). "Will I Be Pretty, Will I Be Rich? Some Thoughts on Theory vs. Practice in Systems Engineering." Third IEEE International Symposium on Requirements Engineering, Annapolis, Maryland, USA, IEEE Computer Society. pp. 184-186.
- Harkness, W. L., W. J. Kettinger, and A. H. Segars (1996). "Sustaining Process Improvement and Innovation in the Information Services Function: Lessons Learned at the Bose Corporation." MIS Quarterly **20**(3): 349-368.
- Harwell, R., E. Aslaksen, I. F. Hooks, R. Mengot, and K. Ptack (1993). "What Is a Requirement." Third Annual International Symposium: Systems Engineering in the Work Place, Arlington, Virginia, USA, International Council on Systems Engineering. pp. 17-24.
- Heidegger, M. (1927). Being and Time. J. Macquerrie and E. Robinson, Oxford, Basil Blackwell.
- Heninger, K. L. (1980). "Specifying Software Requirements for Complex Systems: New Techniques and Their Application." IEEE Transactions on Software Engineering **6**(1): 2-13.
- Hickey, A. M. and A. M. Davis (2003a). "Elicitation Technique Selection: How Do Experts Do It?" 11th IEEE International Requirements Engineering Conference, Monterey Bay, California, USA, IEEE Computer Society. pp. 169- 178.

- Hickey, A. M. and A. M. Davis (2003b). "Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes." 36th Hawaii International Conference on System Sciences, Hilton Waikoloa Village, Hawaii, IEEE Computer Society. pp. 96-105.
- Hofmann, H. F. and F. Lehner (2001). "Requirements Engineering as a Success Factor in Software Projects." IEEE Software **18**(4): 58-66.
- Hooks, I. F. and K. A. Farry (2000). Customer-Centered Products: Creating Successful Products Through Smart Requirements Management. New York, AMACOM.
- Hornby, C., C. Clegg, J. Robson, C. McClaren, S. Richardson, and P. O'Brien (1992). "Human and Organizational Issues in Information Systems Development." Behavior & Information Technology **11**(3): 160-174.
- Hsia, P., A. M. Davis, and D. C. Kung (1993). "Status Report: Requirements Engineering." IEEE Software **10**(6): 75-79.
- Huisman, M. and J. Iivari (2002). "The Individual Deployment of Systems Development Methodologies." The Fourteenth International Conference on Advanced Information Systems Engineering, Lecture Notes in Computer Science 2348, Toronto, Canada, Springer-Verlag. pp. 134-150.
- Huizing, A., E. Koster, and W. Bouman (1997). "Balance in Business Reengineering: An Empirical Study of Fit and Performance." Journal of Management Information Systems **14**(1): 93-118.
- Humphrey, W. S. (1989). Managing the Software Process. Reading, Massachusetts, Addison-Wesley.
- Humphrey, W. S. (1998). "Why Don't They Practice What We Preach?" Annals of Software Engineering **6**(1-4): 201-222.
- Humphrey, W. S. (2002). "Three Process Perspectives: Organization, Teams, and People." Annals of Software Engineering **14**(1-4): 39-72.
- ICRE (1996). Proceedings of the Second IEEE International Conference on Requirements Engineering. Colorado Springs, Colorado, IEEE Computer Society.
- ICRE (1998). Proceedings of the Third IEEE International Conference on Requirements Engineering. Colorado Springs, Colorado, IEEE Computer Society.
- IEEE Standard 610.12 (1990). IEEE Standard 610.12-1990. IEEE Standard for Software Configuration Management Plans. The Institute of Electrical and Electronics Engineers Ed. New York, NY, IEEE Computer Society Press.
- IEEE/EIA 12207.0 (1996). IEEE/EIA 12207.0, Standard for Information Technology - Software life cycle processes. Industry Implementation of International Standard ISO/IEC 12207 : 1995. The Institute of Electrical and Electronics Engineers and Electronic Industries Association Engineering Department Eds. New York, NY, IEEE Computer Society Press.
- IEEE/EIA 12207.1-1997 (1997). IEEE/EIA 12207.1, Guide for ISO/IEC 12207, Standard for Information Technology - Software life cycle processes - Life cycle data. Industry Implementation of International Standard ISO/IEC 12207 : 1995. The Institute of Electrical and Electronics Engineers and Electronic Industries Association Engineering Department Eds. New York, NY, IEEE Computer Society Press.
- IEEE/EIA 12207.1 (1997). IEEE/EIA 12207.1, Guide for ISO/IEC 12207, Standard for Information Technology - Software life cycle processes - Life cycle data. Industry

- Implementation of International Standard ISO/IEC 12207 : 1995. The Institute of Electrical and Electronics Engineers and Electronic Industries Association Engineering Department Eds. New York, NY, IEEE Computer Society Press.
- Iivari, J. (1983). "Contributions to the Theoretical Foundations of Systemeering Research and the PICO Model." Dissertation, Institute of Data Processing, Faculty of Science. Oulu, University of Oulu: 310 p.
- Iivari, J. (1987). "A Hierarchical Spiral Model for the Software Process: Notes on Boehm's Spiral Model." ACM SIGSOFT Software Engineering Notes **12**(1): 35-37.
- Iivari, J. (1990). "Hierarchical Spiral Model for Information System and Software Development. Part 2: Design Process." Information and Software Technology **32**(7): 450-458.
- INCOSE (2003). Tools Survey: Requirements Management (RM) Tools, International Council on Systems Engineering. <http://www.incose.org/tools/tooltax.html>. Accessed March 26, 2003.
- Introna, L. D. and E. A. Whitley (1997). "Against Method-ism: Exploring the Limits of Method." Logistics Information Management **10**(5): 235-245.
- Isacson, P., G. Pedersen, and S. Bang (2001). "Accelerating CMM-Based Improvement Programs: the Accelerator Model and Method with Experiences." Software Process Improvement and Practice **6**(1): 23-34.
- ISO/IEC 9126-1 (2001). Software Engineering - Product Quality - Part 1: Quality Model. ISO/IEC Ed., International Organization for Standardization and International Electrotechnical Commission.
- ISO/IEC 12119 (1994). Information technology - Software packages - Quality requirements and testing. ISO/IEC Ed., International Organization for Standardization and International Electrotechnical Commission.
- ISO/IEC TR 15504-1 (1998). Information Technology - Software Process Assessment - Part 1: Concepts and Introductory Guide. ISO/IEC Ed., International Organization for Standardization and International Electrotechnical Commission.
- ISO/IEC TR 15504-2 (1998). Information Technology - Software Process Assessment - Part 2: A Reference Model for Processes and Process Capability. ISO/IEC Ed., International Organization for Standardization and International Electrotechnical Commission.
- ISO/IEC TR 15504-5 (1998). Information Technology - Software Process Assessment - Part 5: An Assessment Model and Indicator Guidance. ISO/IEC Ed., International Organization for Standardization and International Electrotechnical Commission.
- ISO/IEC TR 15504-9 (1998). Information Technology - Software Process Assessment - Part 9: Vocabulary. ISO/IEC Ed., International Organization for Standardization and International Electrotechnical Commission.
- Jackson, M. A. (1995). Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices. Addison-Wesley.
- Jackson, M. A. (2001). Problem Frames: Analyzing and Structuring Software Development Problems. Addison-Wesley.
- Jacobson, I., G. Booch, and J. Rumbaugh (1998). The Unified Software Development Process. Reading, Massachusetts, Addison-Wesley.
- Jalote, P. (1991). An Integrated Approach to Software Engineering. New York, Springer-Verlag.

- James, L. (1996). "Viewpoint: What's Wrong with Requirements Management Tools?" Requirements Engineering **1**(3): 190-194.
- Jeffery, R. and M. Berry (1993). "A Framework for Evaluation and Prediction of Metrics Program Success." First International Software Metrics Symposium, Baltimore, MD. pp. 28-39.
- Jones, T. C. (1996). Applied Software Measurement: Assuring Productivity and Quality. McGraw-Hill.
- Juristo, N., A. M. Moreno, and A. Silva (2002). "Is the European Industry Moving toward Solving Requirements Engineering Problems?" IEEE Software **19**(6): 70-77.
- Järvinen, P. (1999). On Research Methods. Tampere, Finland, Opinpaja Oy.
- Kaindl, H. (2000). "Why Is It So Difficult to Introduce Requirements Engineering Research Results into Mainstream Requirements Engineering Practice." Fourth International Conference on Requirements Engineering, Schaumburg, Illinois, IEEE Computer Society Press. pp. 67-68.
- Kaindl, H., S. Brinkkemper, J. A. J. Bubenko, B. Farbey, S. J. Greenspan, C. L. Heitmeyer, J. C. S. d. P. Leite, N. R. Mead, J. Mylopoulos, and J. Siddiqi (2002). "Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda." Requirements Engineering **7**(3): 113-123.
- Kaltio, T. and A. Kinnula (2000). "Deploying the Defined SW Process." Software Process Improvement and Practice **5**(1): 65-83.
- Kamsties, E., K. Hörmann, and M. Schlich (1998). "Requirements Engineering in Small and Medium Enterprises." Requirements Engineering **3**(2): 84-90.
- Kauppinen, M. and S. Kujala (2001). "Assessing Requirements Engineering Processes with the REAIMS Model: Lessons Learned." Eleventh Annual International Symposium of the International Council on Systems Engineering, Melbourne, Australia, INCOSE.
- Kauppinen, M., S. Kujala, T. Aaltio, and L. Lehtola (2002). "Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice." IEEE Joint International Conference on Requirements Engineering, Essen, Germany, IEEE Computer Society. pp. 43-51.
- Kautz, K., H. W. Hansen, and K. Thaysen (2000). "Applying and Adjusting a Software Process Improvement Model in Practice: The Use of the IDEAL Model in a Small Software Enterprise." 22nd International Conference on Software Engineering, Limerick, Ireland, IEEE Computer Society. pp. 626 - 633.
- Keil, M., P. E. Cule, K. Lyytinen, and R. C. Schmidt (1998). "A Framework for Identifying Software Project Risks." Communications of the ACM **41**(11): 76-83.
- Kerola, P. and P. Järvinen (1975). Systemointi II: Tietosysteemin rakentamisen ja käytön systeemitoreettinen malli (In Finnish). Helsinki, Oy Gaudeamus Ab.
- Kettinger, W. J., J. T. C. Teng, and S. Guha (1997). "Business Process Change: A Study of Methodologies, Techniques, and Tools." MIS Quarterly **21**(1): 55-80.
- Kilpi, T. (1997). "Product Management Challenge to Software Change Process: Preliminary Results from Three SMEs Experiment." Software Process Improvement and Practice **3**(3): 165-175.
- Kitchenham, B. A. (1996). "Evaluating Software Engineering Methods and Tool Part 1: The Evaluation Context and Evaluation Methods." Software Engineering Notes **21**(1): 11-15.

- Kitchenham, B. A., L. M. Pickard, and S. L. Pfleeger (1995). "Case Studies for Method and Tool Evaluation." IEEE Software **12**(4): 52-62.
- Kolb, D. A. and A. L. Frohman (1970). "An Organizational Development Approach to Consulting." Sloan Management Review **12**(1): 51-65.
- Kotonya, G. (1999). "Practical Experience with Viewpoint-Oriented Requirements Specification." Requirements Engineering **4**(3): 115-133.
- Kotonya, G. and I. Sommerville (1996). "Requirements Engineering with Viewpoints." Software Engineering Journal **11**(1): 5-11.
- Kotonya, G. and I. Sommerville (1997). Requirements Engineering: Processes and Techniques - Homepage. <http://www.comp.lancs.ac.uk/computing/resources/re/>. Accessed August 5, 2002.
- Kotonya, G. and I. Sommerville (1998). Requirements Engineering: Processes and Techniques. Chichester, John Wiley & Sons.
- Kovitz, B. M. (1998). Practical Software Requirements: A Manual of Content and Style. Greenwich, Connecticut, USA, Manning Publishing Company.
- Krogstie, J. (1999). "Using Quality Function Deployment in Software Requirements Specification." The Fifth International Workshop on Requirements Engineering: Foundations for Software Quality, Heidelberg, Germany. pp. 171-185.
- Kulak, D. and E. Guiney (2000). Use Cases: Requirements In Context. Boston, Addison-Wesley.
- Kumar, K. and R. J. Welke (1992). Methodology Engineering: A Proposal for Situation-Specific Methodology Construction. Challenges and Strategies for Research in Systems Development. W. W. Cotterman and J. A. Senn Eds. Chichester, John Wiley & Sons. pp. 257-269.
- Kuvaja, P., P. Jorma, and A. Bicego (1999a). "TAPISTRY - A Software Process Improvement Approach Tailored for Small Enterprises." Software Quality Journal **8**(2): 149-156.
- Kuvaja, P., S. Jouni, K. Krzanik, A. Bicego, G. Koch, and S. Saukkonen (1994). Software Processes Assessment and Improvement: The BOOTSTRAP Approach. U.K., Blackwell Publishers.
- Kuvaja, P., J. Palo, and A. Bicego (1999b). "TAPISTRY - A Software Process Improvement Approach Tailored for Small Enterprises." Software Quality Journal **8**(2): 149-156.
- LaBudde, E. V. (1997). "Designer's Toolbox: Reducing Development Time with Requirements Management." Medical Device & Diagnostic Industry Magazine.
- Laitinen, M., M. E. Fayad, and R. P. Ward (2000). "The Problem with Scalability." Communications of the ACM **43**(9): 105-107.
- Laplante, P. A., C. J. Neill, and C. Jacobs (2002). "Software Requirements Practices: Some Real Data." 27th Annual NASA Goddard/IEEE Software Engineering Workshop, IEEE Computer Society. pp. 121-128.
- Lauesen, S. (2002). Software Requirements - Styles and Techniques. Harlow, Addison-Wesley.
- Lee, J. and N.-L. Xue (1999). "Analyzing User Requirements by Use Cases: A Goal-Driven Approach." IEEE Software **16**(4): 92-101.
- Leffingwell, D. and D. Widrig (1999). Managing Software Requirements: A Unified Approach. Reading, Massachusetts, Addison-Wesley.
- Liu, X. F. (2000). "Software Quality Function Deployment." IEEE Potentials **19**(5): 14-16.

- Lubars, M., C. Potts, and C. Richter (1993). "A Review of the State of the Practice in Requirements Modeling." IEEE International Symposium on Requirements Engineering, San Diego, CA, IEEE Computer Society. pp. 2-14.
- Lukka, K. (2002). The Constructive Research Approach. <http://www.metodix.com>. Accessed July 24, 2002.
- Lycett, M., R. D. Macredie, C. Patel, and R. J. Paul (2003). "Migrating Agile Methods to Standardized Development Practice." Computer **36**(6): 79-85.
- Maansaari, J. and J. Iivari (1999). "The Evolution of CASE Usage in Finland Between 1993 and 1996." Information and Management **36**(1): 37-53.
- Maciaszek, L. A. (2001). Requirements Analysis and System Design: Developing Information Systems with UML. Harlow, England, Addison-Wesley.
- Mackay, J. M. and J. J. Elam (1992). "A Comparative Study on How Experts and Novices Use a Decision Aid to Solve Problems in Complex Knowledge Domains." Information Systems Research **3**(2): 150-172.
- Mackay, W. E. (1990). "Users and Customizable Software: A Co-Adaptive Phenomenon." Doctoral dissertation, Alfred P. Sloan School of Management. Cambridge, Massachusetts, USA, Massachusetts Institute of Technology: 202 p.
- Mackay, W. E. (1991). "Triggers and Barriers to Customizing Software." The SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology, New Orleans, Louisiana, USA, ACM Press. pp. 153-160.
- Maiden, N. A. M. and G. Rugg (1996). "ACRE: Selecting Methods for Requirements Acquisition." Software Engineering Journal **11**(3): 183-192.
- Mannio, M. and U. Nikula (2001). "Requirements Elicitation Using a Combination of Prototypes and Scenarios." IV Workshop on Requirements Engineering, Buenos Aires, Argentina, Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Argentina. pp. 283-296.
- March, S. T. and G. F. Smith (1995). "Design and Natural Science Research on Information Technology." Decision Support Systems **15**(4): 251-266.
- Marciniak, J. J. (2002). Encyclopedia of Software Engineering. 2nd edition. New York, John Wiley & Sons.
- Matulevicius, R. (2004). "How Requirements Specification Quality Depends on Tools: A Case Study." The 16th International Conference on Advanced Information Systems Engineering, Riga, Latvia, Springer-Verlag. pp. 353-367.
- McAuley, J. E. (1992). "What Should Systems Engineers Know and When Should They Know It." Second Annual International Symposium: Systems Engineering for the 21st Century, Seattle, Washington, USA, International Council on Systems Engineering. pp. 11-18.
- McFeeley, B. (1996). IDEAL: A User's Guide for Software Process Improvement. Handbook CMU/SEI-96-HB-001. Pittsburgh, PE, USA, Software Engineering Institute, Carnegie Mellon University: 236 p. Available at <http://www.sei.cmu.edu/publications/documents/96.reports/96.hb.001.html>.
- McGraw, K. and K. Harbison (1997). User-Centered Requirements: The Scenario-Based Engineering Process. Mahwah, New Jersey, Lawrence Erlbaum Associates.
- McPhee, C. and A. Eberlein (2002). "Requirements Engineering for Time-to-Market Projects." Ninth IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Lund, Sweden, IEEE Computer Society. pp. 17-24.

- Mead, N. (2000). "Requirements Management and Requirements Engineering: You Can't Have One Without The Other." Cutter IT Journal **13**(5): 4-8.
- Meilich, A. (1997). "Lessons Learned from Use of a Computer-Based Requirements Management Tool in the Project Configuration Control Board Environment." Seventh Annual International Symposium: Systems Engineering: A Necessary Science, Los Angeles, CA, USA, International Council on Systems Engineering.
- Menzies, T., S. Easterbrook, B. Nuseibeh, and S. Waugh (1999). "An Empirical Investigation of Multiple Viewpoint Reasoning in Requirements Engineering." IEEE International Symposium on Requirements Engineering, Limerick, Ireland, IEEE Computer Society. pp. 100-109.
- Miller, S. (1997). "How Can Requirements Engineering Research Become Requirements Engineering Practice." Third IEEE International Symposium on Requirements Engineering, Annapolis, Maryland, USA, IEEE Computer Society Press. pp. 260.
- Moore, G. A. (1999). Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers. Revised edition. New York, HarperCollinsPublishers.
- Moore, G. C. and I. Benbasat (1991). "Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation." Information Systems Research **2**(3): 192-222.
- Morris, P., M. Masera, and M. Wilikens (1998). "Requirements Engineering and Industrial Uptake." Requirements Engineering **3**(2): 79-83.
- National Competitiveness Council (1999). Annual Competitiveness Report. Dublin, Ireland. Available at <http://www.forfas.ie/ncc/reports.html>.
- Nawrocki, J., M. Jasinski, B. Walter, and A. Wojciechowski (2002). "Extreme Programming Modified: Embrace Requirements Engineering Practices." IEEE Joint International Conference on Requirements Engineering, Essen, Germany, IEEE Computer Society. pp. 303-310.
- Neill, C. J. and P. A. Laplante (2003). "Requirements Engineering: The State of the Practice." IEEE Software **20**(6): 40-45.
- Newell, A. (1969). Heuristic Programming: Ill-Structured Problems. Progress in Operations Research: Relationship Between Operations Research and the Computer. J. S. Aronofsky Ed. New York, John Wiley & Sons. Vol III, pp. 360-414.
- Nidumolu, S. R. (1996). "A Comparison of the Structural Contingency and Risk-Based Perspectives on Coordination in Software-Development Projects." Journal of Management Information Systems **13**(2): 77-113.
- Nidumolu, S. R. and G. W. Knotts (1998). "The Effects of Customizability and Reusability on Perceived Process and Competitive Performance of Software Firms." MIS Quarterly **22**(2): 105-137.
- Nikula, U. (2002a). "BaRE - A Ready to Use Method for Requirements Engineering." Licentiate Thesis, Department of Information Technology. Lappeenranta, Lappeenranta University of Technology: 77 p.
- Nikula, U. (2002b). The BaRE Method Version 1.0. Department of Information Technology Report 3. Lappeenranta, Lappeenranta University of Technology: 150 p.
- Nikula, U. (2003a). "Experiences from Lightweight RE Method Evaluations." Comparative Evaluation in Requirements Engineering, Monterey Bay, California, USA, IEEE Computer Society. pp. 53-60.

- Nikula, U. (2003b). "Experiences of a Ready-to-Use Requirements Documentation Set." Eighth Australian Workshop on Requirements Engineering, Sydney, Australia, Faculty of Information Technology, University of Technology, Sydney. pp. 43-52.
- Nikula, U. and J. Sajaniemi (2002). "BaSyRE: A Lightweight Combination of Proven RE Techniques." International Workshop on Time-Constrained Requirements Engineering, Essen, Germany, Paper Virtual Editora, Rio de Janeiro. pp. 69-78.
- Nikula, U., J. Sajaniemi, and H. Kälviäinen (2000a). "Management View on Current Requirements Engineering Practices in Small and Medium Enterprises." The Fifth Australian Workshop on Requirements Engineering, Brisbane, Australia, Faculty of Information Technology, University of Technology, Sydney. pp. 81-89.
- Nikula, U., J. Sajaniemi, and H. Kälviäinen (2000b). A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises. Research Report 1. Lappeenranta, Finland, Telecom Business Research Center Lappeenranta: 26 p. Available at <http://www.tbrc.fi/>.
- Nikula, U., J. Sajaniemi, and H. Kälviäinen (2000c). "A State-of-the-Practice Survey on Requirements Engineering: Industry Expectations on Education and Technology Transfer." Fifth International Conference on Software Process Improvement Research, Education and Training - INSPIRE V, London, Great Britain, The British Computer Society. pp. 13-24.
- Nishiyama, T., K. Ikeda, and T. Niwa (2000). "Technology Transfer Macro-Process: A Practical Guide for the Effective Introduction of Technology." The 22nd International Conference on Software Engineering, Limerick, Ireland, ACM Press. pp. 577-586.
- Norman, D. A. (1990). The Design of Everyday Things. London, England, The MIT Press.
- Nunamaker, J. F., Jr, C. Minder, and T. D. M. Purdin (1990). "Systems Development in Information Systems Research." Journal of Management Information Systems 7(3): 89-106.
- Nunnally, J. C. and I. A. Bernstein (1994). Psychometric Theory. 3rd edition. New York, McGraw-Hill.
- Nuseibeh, B. (2001). "Weaving Together Requirements and Architectures." Computer 34(3): 115-117.
- Nuseibeh, B. and S. Easterbrook (2000). "Requirements Engineering: A Roadmap." International Conference on Software Engineering, The Future of Software Engineering, 2nd Edited Volume, Limerick, Ireland, ACM Press. pp. 35-46.
- Odell, J. J. (1996). "A Primer to Method Engineering." Method Engineering: Principles of Method Construction and Tool Support, Atlanta, USA, Chapman & Hall. pp. 1-7.
- Oliver, D. (1995). "Systems Engineering & Software Engineering, Contrasts and Synergism." Fifth Annual International Symposium: Systems Engineering in the Global Market Place, St. Louis, Missouri, USA, International Council on Systems Engineering.
- Oxford University Press (1989). Oxford English Dictionary. <http://dictionary.oed.com>. Accessed September 20, 2004. Second edition.
- Parnas, D. L. and P. C. Clements (1986). "A Rational Design Process: How and Why to Fake It." IEEE Transactions on Software Engineering 12(2): 251-257.
- Parnas, D. L. and J. Madey (1995). "Functional Documents for Computer Systems." Science of Computer Programming 25(1): 41-61.

- Paton, G. (1992). "SSM as an Aid to Requirements Definition." International Conference on Information Decision Action Systems in Complex Organizations, IEEE Computer Society. pp. 90-94.
- Pfleeger, S. L. (1994). "Design and Analysis in Software Engineering Part 1: The Language of Case Studies and Formal Experiments." Software Engineering Notes **19**(4): 16-20.
- Pfleeger, S. L. (1998). Software Engineering: Theory and Practice. Upper Saddle River, New Jersey, Prentice Hall.
- Pohl, K. (1994). "The Three Dimensions of Requirements Engineering: A Framework and Its Applications." Information Systems **19**(3): 243-258.
- Port, D. and E. Dincel (2001). "Experiences Using Domain Specific Techniques within Multimedia Software Engineering." Annals of Software Engineering **12**(1): 11-45.
- Pressman, R. S. (1998). Software Engineering: A Practitioner's Approach. 4th edition, European Adaptation. New York, McGraw-Hill.
- Pressman, R. S. (2001a). Adaptable Process Model Document Templates. <http://www.rspa.com/docs/>. Accessed March 6, 2002.
- Pressman, R. S. (2001b). Software Engineering: A Practitioner's Approach. 5th edition. Boston, McGraw-Hill.
- Probert, S. K. (1999). "Requirements Engineering, Soft Systems Methodology and Workforce Empowerment." Requirements Engineering **4**(2): 85-91.
- Ramesh, B., J. Pries-Heje, and R. L. Baskerville (2002). "Internet Software Engineering: A Different Class of Processes." Annals of Software Engineering **14**(1-4): 169-195.
- Rational Software Corporation (2002). Rational Unified Process Roadmap: Small Projects, Rational Software Corporation. CD-ROM version 2002.05.00.
- RE (1997). Proceedings of the Third IEEE International Symposium on Requirements Engineering. Annapolis, Maryland, USA, IEEE Computer Society.
- RE (1999). Proceedings of the Fourth IEEE International Symposium on Requirements Engineering. University of Limerick, Ireland, IEEE Computer Society.
- Redwine, S. T., Jr and W. E. Riddle (1985). "Software Technology Maturation." The 8th International Conference on Software Engineering, London, England, ACM Press. pp. 189-200.
- Reels, J. S. (1999). "Critical Success Factors in Software Projects." IEEE Software **16**(3): 18-23.
- Regnell, B., P. Beremark, and O. Eklundh (1998). "A Market Driven Requirements Engineering Process: Results from an Industrial Process Improvement Programme." Requirements Engineering **3**(2): 121-129.
- Reinikainen, L., U. Nikula, K. Leino, M. Tuominen, and H. Kälviäinen (2001). "Elicitation of Customer Requirements with Group Methods in Software Engineering." Challenges of Innovation and Technology Management for the New Millennium, Lappeenranta University of Technology, Finland, Department of Industrial Engineering and Management, Lappeenranta University of Technology, Finland. pp. 302-310.
- Richardson, I. (2002). "SPI Models: What Characteristics are Required for Small Software Development Companies?" Software Quality Journal **10**(2): 101-114.
- Riddle, W. E. and R. E. Fairley (1980). Software Development Tools. Berlin, Springer-Verlag.

- Riemenschneider, C. K., B. C. Hardgrave, and F. D. Davis (2002). "Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models." IEEE Transactions on Software Engineering **28**(12): 1135-1145.
- Riemenschneider, C. K., D. A. Harrison, and P. P. J. Mykytyn (2003). "Understanding IT Adoption Decisions in Small Business: Integrating Current Theories." Information and Management **40**(4): 269-285.
- Robertson, S. and J. Robertson (1999). Mastering the Requirements Process. Harlow, England, Addison-Wesley.
- Robertson, S. and J. Robertson (2001). Volere Requirements Specification Template, Edition 8, Atlantic Systems Guild. www.systemsguild.com. Accessed November 29, 2001.
- Rogers, E. M. (1995). Diffusion of Innovations. 4th edition. New York, The Free Press.
- Rolland, C. and G. Grosz (1994). "A General Framework for Describing the Requirements Engineering Process." IEEE International Conference on Systems, Man, and Cybernetics, 1, IEEE Computer Society. pp. 818-823.
- Rolland, C., N. Prakash, and A. Benjamin (1999). "A Multi-Model View of Process Modelling." Requirements Engineering **4**(4): 169-187.
- Saastamoinen, I. and M. Tukiainen (2003a). Company A: Assessment Report (Confidential). Joensuu, University of Joensuu: 73 p.
- Saastamoinen, I. and M. Tukiainen (2003b). Company B: Assessment Report (Confidential). Joensuu, University of Joensuu: 70 p.
- Saastamoinen, I. and M. Tukiainen (2003c). Company C: Assessment Report (Confidential). Joensuu, University of Joensuu: 69 p.
- Sawyer, P., I. Sommerville, and S. Viller (1999). "Capturing the Benefits of Requirements Engineering." IEEE Software **16**(2): 78-85.
- Scharer, L. (1981). "Pinpointing Requirements." Datamation **27**(4): 139-151.
- Scientific Software Development (2004). ATLAS.ti - The Knowledge Workbench, Scientific Software Development. <http://www.atlasti.de/>. Accessed April 22, 2004.
- SCRUM (2002). SCRUM Development Process Home Page. www.controlchaos.com. Accessed July 18, 2002.
- sfrm (2002). Software for Requirements Management. http://cs.joensuu.fi/pages/saja/se/sfrm_mainnos.html. Accessed June 17, 2002.
- Siddiqi, J. and M. C. Shekaran (1996). "Requirements Engineering: The Emerging Wisdom." IEEE Software **13**(2): 15-19.
- Software Engineering Institute (1995). The Capability Maturity Model: Guidelines for Improving the Software Process. Reading, MA, Addison-Wesley.
- Software Engineering Institute (2002). Capability Maturity Models. <http://www.sei.cmu.edu/cmm/cmms/cmms.html>. Accessed 4 July 2002.
- Sommerville, I. (1998). Software Engineering. 5th edition. Wokingham, England, Addison-Wesley.
- Sommerville, I. (2001). Software Engineering. 6th edition. Harlow, England, Addison-Wesley.
- Sommerville, I. and P. Sawyer (1997). Requirements Engineering: A Good Practice Guide. Chichester, England, John Wiley & Sons.

- Sommerville, I., P. Sawyer, and S. Viller (1999). "Managing Process Inconsistency Using Viewpoints." IEEE Transactions on Software Engineering **25**(6): 784-799.
- SOPHIST Group (2002). SOPHIST - Group for Innovative Software Engineering Ltd., SOPHIST Group. www.sophist.de. Accessed September 25, 2002.
- Straub, D. W. (1989). "Validating Instruments in MIS Research." MIS Quarterly **13**(2): 147-169.
- STRÍ TS2 (1991). Modelling a Software Quality Handbook. Reykjavik, Iceland, Icelandic Council for Standardization (STRÍ).
- Suchman, L. (1989). Notes on Computer Support for Cooperative Work. Working Paper WP-12. Jyväskylä, University of Jyväskylä: 18 p.
- Swartout, W. and R. Balzer (1982). "On the Inevitable Intertwining of Specification and Implementation." Communications of the ACM **25**(7): 438-440.
- Sweeney, A. and D. W. Bustard (1997). "Software Process Improvement: Making It Happen in Practice." Software Quality Journal **6**(4): 265-273.
- Söderström, T. and P. Stoica (1989). System Identification. Hertfordshire, Prentice Hall.
- Tamai, T. (1993). "Current Practices in Software Processes for System Planning and Requirements Analysis." Information and Software Technology **35**(6/7): 339-344.
- ter Hofstede, A. H. M. and T. P. van der Weide (1992). "Formalization of Techniques: Chopping Down the Methodology Jungle." Information and Software Technology **34**(1): 57-65.
- ter Hofstede, A. H. M. and T. F. Verhoef (1997). "On the Feasibility of Situational Method Engineering." Information Systems **22**(6-7): 401-422.
- Thayer, R. H. (1997). Software System Engineering: An Engineering Process. Software Requirements Engineering. R. H. Thayer and M. Dorfman Eds, IEEE Computer Society Press. pp. 84-106.
- Thayer, R. H. and M. Dorfman, Eds. (1997). Software Requirements Engineering. Los Alamitos, California, IEEE Computer Society Press.
- The European Commission (2003). SME Definition, The European Commission. http://europa.eu.int/comm/enterprise/enterprise_policy/sme_definition/index_en.htm. Accessed 8 March 2005.
- The Standish Group (1994). The CHAOS Report, The Standish Group. www.standishgroup.com. Accessed August 16, 2004.
- The Standish Group (2001). "CHAOS Chronicles II. pp. 300.
- The White House (2001). The State of the Small Business: A Report of the President 1999-2000. Washington, The White House Together with the Office of Advocacy's Annual Report on Small Business and Competition: 262 p. Available at http://www.sba.gov/advo/stats/stateofsb99_00.pdf.
- Thompson, R., C. Higgins, and J. Howell (1991). "Personal Computing: Toward a Conceptual Model of Utilization." MIS Quarterly **15**(1): 125-143.
- Thong, J. Y. L. (1999). "An Integrated Model of Information Systems Adoption in Small Businesses." Journal of Management Information Systems **15**(4): 187-214.
- Thong, J. Y. L., C. Yap, and K. Raman (1996). "Top Management Support, External Expertise and Information Systems Implementation in Small Businesses." Information Systems Research **7**(2): 248-267.

- Tolvanen, J.-P., M. Rossi, and H. Liu (1996). "Method Engineering: Current Research Directions and Implications for Future Research." *Method Engineering: Principles of Method Construction and Tool Support*, Atlanta, USA, Chapman & Hall. pp. 296-317.
- Tornatzky, L. G. and M. Fleischer (1990). The Processes of Technological Innovation. Lexington, MA, Lexington Books.
- Tornatzky, L. G. and K. J. Klein (1982). "Innovation Characteristics and Innovation Adoption-Implementation: A Meta-Analysis of Findings." IEEE Transactions on Engineering Management **EM-29**(1): 28-45.
- Toro, A., Durán (2002). REM main page. <http://klendathu.lsi.us.es/REM/>. Accessed September 25, 2002.
- tSoft (2002). tSoft - Ohjelmistotuotannon Tietokeskus, University of Joensuu, Department of Computer Science. <http://cs.joensuu.fi/pages/saja/tSoft/>. Accessed July 24, 2002.
- Tvete, B. (1999). "Introducing Efficient Requirements Management." Tenth International Workshop on Database & Expert Systems Applications, Florence, Italy, IEEE Computer Society. pp. 370-375.
- United States Small Business Administration (2003). What Is Small Business?, United States Small Business Administration. <http://www.sba.gov/size/>. Accessed November 17, 2003.
- van Lamsweerde, A. (2000a). "Goal-Oriented Requirements Engineering: A Guided Tour." Fifth International Symposium on Requirements Engineering, IEEE Computer Society. pp. 249-262.
- van Lamsweerde, A. (2000b). "Requirements Engineering in the Year 00: A Research Perspective." International Conference on Software Engineering, Limerick, Ireland, ACM Press. pp. 5-19.
- van Schouwen, A. J., D. L. Parnas, and J. Madey (1993). "Documentation of Requirements for Computer Systems." *Proceedings of IEEE International Symposium on Requirements Engineering*. pp. 198 -207.
- Ward, R. P., M. E. Fayad, and M. Laitinen (2001). "Software Process Improvement in the Small." Communications of the ACM **44**(4): 105-107.
- Watkins, J. (2001). Testing IT: An Off-the-Shelf Software Testing Process. Cambridge, Cambridge University Press.
- Weidenhaupt, K., K. Pohl, M. Jarke, and P. Haumer (1998). "Scenarios in Systems Development: Current Practice." IEEE Software **14**(2): 34-45.
- Weinberg, G. M. (1988). Rethinking Systems Analysis & Design. New York, Dorset House Publishing.
- Weinberg, G. M. and E. L. Schulman (1974). "Goals and Performance in Computer Programming." Human Factors **16**(1): 70-77.
- Venkatesh, V. and F. D. Davis (2000). "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies." Management Science **46**(2): 186-204.
- Vessey, I. (1991). "Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature." Decision Sciences **22**(2): 219-240.
- Vessey, I. (1997). "Problems Versus Solutions: The Role of the Application Domain in Software." The Seventh Workshop on Empirical Studies of Programmers, Alexandria, Virginia, USA. pp. 233-240.

- Vessey, I. and R. L. Glass (1998). "Strong vs. Weak Approaches to Systems Development." Communications of the ACM **41**(4): 99-102.
- Vessey, I. and R. Weber (1986). "Structured Tools and Conditional Logic: An Empirical Investigation." Communications of the ACM **29**(1): 48-57.
- White, S., M. Alford, J. Holzman, S. Kuehl, B. McCay, D. Oliver, D. Owens, C. Tully, and A. Willey (1993). "Systems Engineering of Computer-Based Systems." Computer **26**(22): 54-65.
- Vickers, A., A. Mavin, and H. May (2002). "Requirements Engineering: How Do You Know How Good You Are?" IEEE Joint International Requirements Engineering Conference, Industrial Presentations, University of Essen, Germany, IEEE Computer Society. pp. 91-99.
- Wieggers, K. E. (1999). Software Requirements. Redmond, Washington, Microsoft Press.
- Wieringa, R. J. (2003). "Methodologies of Requirements Engineering Research and Practice: Position Statement." Comparative Evaluation in Requirements Engineering, Monterey Bay, California, USA, IEEE Computer Society. pp. 7-11.
- Wilson, D. N., T. Hall, and N. Baddoo (2001). "A Framework for Evaluation and Prediction of Software Process Improvement Success." Journal of Systems and Software **59**(2): 135-142.
- Winograd, T. and F. Flores (1986). Understanding Computers and Cognition: A New Foundation for Design. Norwood, NJ, Ablex Publishing Corporation.
- Yamamoto, S., H. Tadaumi, and M. Ueno (1996). "DREM: Domain-Based Requirements Engineering Methodology." NTT R&D **45**(8): 711-718.
- Yin, R. K. (2002). Case Study Research: Design and Methods. 3rd edition. Thousand Oaks, CA, SAGE Publications.
- Young, R. R. (2001). Effective Requirements Practices. Boston, Addison-Wesley.
- Zahran, S. (1998). Software Process Improvement: Practical Guidelines for Business Success. Harlow, England, Addison-Wesley.
- Zave, P. (1997). "Classification of Research Efforts in Requirements Engineering." ACM Computing Surveys **29**(4): 315-320.
- Zelkowitz, M. V. and D. R. Wallace (1998). "Experimental Models for Validating Technology." Computer **31**(5): 23-31.
- Zowghi, D. (2002). Requirements Engineering Online. <http://www-staff.it.uts.edu.au/~didar/RE-online.html>. Accessed July 17, 2002.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
26	Requirements Document Template Comparison																	
27	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Alt	H	I	J	K	L	M	N	O	Code	Notes	
28	General Document Information																2	
29	Document Title										x	x	x	x	x		2	
30	Document Number										x						0	If relevant
31	Date: Created, Issued, Last Modified, Date									x	x		x	x			1	
32	Status									x							2	
33	Issuing Organization									x	x	x	x				2	
34	Document Preparer(s)										x	x	x	x			2	
35	Project Name											x	x				2	
36	Project Billing Code										x						0	If relevant
37	Proprietary and Confidential -words as appropriate										x						1	
38	Document Version											x	x				2	
39	Change History									x	x	x	x				2	
40	Approval Authority									x							0	If relevant
41	Copyright Notice											x					0	
42	Table of Contents								x		x	x	x				2	
43	Introduction								x			x	x				2	
44	Document Purpose								x			x	x				-1	Standard purpose
45	Product Identification									x	x	x	x				2	
46	Customer for the Software										x						2	
47	Product Purpose											x					2	
48	Product Scope								x	x	x	x	x	x			2	Handled as Product Purpose, Position, Goals &Context
49	Product Position Statement											x		x			2	
50	Vision Statement																	Covered in Product Position Statement & Purpose
51	Project Issues												x				3	
52	Project Priorities												x				-2	Project plan*
													x				-2	Project plan*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
53			Project Success Factors										x				-2	Project plan*
54		Business Requirements											x				3	Business Goals*
55		Background										x	x				1	
56		Business Opportunity										x	x				-1	Product Position Statement*
57		Alternatives and Competition										x					1	
58		Business Objectives									x	x	x				-1	Business Goals*
59		Customer/Market Requirements									x		x				-1	Business Goals* or Detailed Requirements*
60		Value Provided to Customers											x				-1	Product Position Statement*
61		Business Risks											x				-1	Suits better to business documents
62		Definitions								x	x			x			3	Glossary*
63		Acronyms								x				x			3	Glossary*
64		Abbreviations								x							3	Glossary*
65		Document Conventions									x		x				-1	If relevant
66		Typographical Conventions									x		x				-1	If relevant
67		Other Conventions											x				-1	If relevant
68		Document Overview							x	x	x		x				2	
69		Intended Audience							x				x				2	
70		Reading Suggestions											x				2	
71		References							x	x	x	x	x				2	
72		Overall Description							x				x				2	
73		Product Context												x			2	Include for Context Diagram
74		Product Boundary													x		1	Handle without own heading
75		Product Perspective							x			x	x				2	
76		System Interfaces							x								2	
77		User Interfaces							x			x	x				2	
78		Ergonomics								x							-1	If relevant

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
79				Constraints on Personnel					x						-1	If relevant
80				Areas Needing Special Attention					x						-1	If relevant
81				Hardware Interfaces			x		x	x	x				-1	If relevant
82				Software Interfaces			x		x	x	x				-1	System Interfaces*
83				Communications Interfaces			x		x	x					-1	System Interfaces*
84				Memory Constraints			x			x					-1	Typical Computer Configuration*
85				Operation Types			x		x						1	
86				Operating Procedures			x		x	x					3	Scenarios*
87				Feature Attributes							x				-1	If relevant
88				Site Adaptation Requirements			x								-1	If relevant
89				Delivery Instructions					x						-1	If relevant
90				Installation Requirements at the Operations Site					x	x	x				-1	If relevant
91				Acceptance Requirements at Operations Site(s)					x						-1	If relevant
92				Installation Requirements at Maintenance Site(s)					x						-1	If relevant
93				Acceptance Requirements at Maintenance Site(s)					x						-1	If relevant
94				Required Data Modifications and Translations								x			-1	If relevant
95				Product Functions											2	
96				Product Features						x	x				2	
97				Use Cases						x	x				2	
98				Product Functions							x				3	Product Features*
99				Physical Characteristics					x						-1	If relevant
100				Special Usage Requirements											-1	If relevant
101				System Requirements Allocations											2	
102				Scope of Initial Release							x				2	
103				Scope of Subsequent Releases							x				1	
104				Manual Operations					x						2	
105				Concept of Execution					x						-1	If relevant
106				Rationale for Allocations					x						1	
107				Restrictions, Limitations and Exclusions							x	x	x		1	
108				Apportioning of Requirements					x						2	
109				Criticality of Requirements					x						2	Handle without own heading

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
110			Precedence of Requirements							x	x						Handle without own heading
111		Stakeholders											x	x			2
112			Stakeholder Profiles										x	x			2
113		User Descriptions										x					2
114			User Classes/Actors									x	x	x			2
115			Characteristics/Profiles				x					x	x	x			2
116			Environment									x	x				2
117			Priorities										x	x			1
118			Participation to Project											x			-1
119			Administrative Users								x						Interface Specification*
120		Problem-Domain Description									x						1 If relevant
121			Sequences of Events									x					-1 Scenarios*
122			Event Responses									x					-1 Scenarios*
123			Causal Rules								x						Important Domain Properties*
124			Important Domain Properties														If relevant (Relevant Facts and Assumptions and in Volere)
125		Constraints								x			x	x			2
126			Regulatory Policies					x									-1
127			Hardware Limitations					x									-1
128			Parallel Operation					x									-1
129			Audit Functions					x									-1
130			Control Functions					x									-1
131			Higher-Order Language Requirements					x									-1
132			Signal Handshake Protocols					x									-1
133			Criticality of the Application					x									-1
134			Current Implementation Environment											x			-1
135			Partner Applications											x			-1
136			Commercial Off-the-Shelf Packages									x		x			-1
137			Anticipated Workplace Environment											x			-1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
138			Schedule Constraints											x		-1	
139			Financial Constraints									x		x		-1	
140			Price Constraints									x				-1	
141			Safety Requirements							x	x		x	x		2	
142		Assumptions					x					x	x	x		2	
143		Dependencies					x					x	x			2	
144		Specific Requirements														2	
145		External Interface Requirements					x			x	x	x	x	x		2	
146		System Interfaces					x	x								2	
147		User Interfaces					x			x	x	x	x	x		2	
148		Hardware Interfaces					x				x	x	x			-1	System Interfaces*
149		Software Interfaces					x				x	x	x	x		-1	System Interfaces*
150		Communications Interfaces					x	x				x	x			-1	System Interfaces*
151		New Application Specific Communication Protocols									x					-1	If relevant
152		File Formats									x					-1	If relevant
153		Problem Frames									x					-1	†
154		Information Queries									x					-1	†
155		Behavioral Rules									x					-1	†
156		Data Input/Output Mappings									x					-1	†
157		Operations on Realized Domains									x					-1	†
158		Behavioral Model and Description									x					-1	†
159		Description for Software Behavior													x	1	If relevant
160		Events													x	0	If relevant
161		States													x	0	If relevant
162		State Transition Diagrams													x	1	If relevant
163		Control Specification													x	0	If relevant
164		Functional Model and Description													x	0	If relevant
165		Description for Function n													x	0	If relevant
166		Processing Narrative for Function n													x	0	If relevant
167		Function n Flow Diagram													x	0	If relevant
168		Function n Interface Description													x	0	If relevant
169		Function n Transforms													x	0	If relevant
170		Transform k Description													x	0	If relevant

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
171					Transform k Interface Description										x	0	If relevant
172					Transform k Lower Level Flow Diagrams										x	0	If relevant
173				Control Flow Description											x	0	If relevant
174		Functional Requirements						x	x	x		x				2	
175		Functionality							x							0	If relevant
176		Input							x							0	If relevant
177		Transformation							x							0	If relevant
178		Output							x							0	If relevant
179		Mode					1	x								0	If relevant
180				Functional Requirement			1	x								0	If relevant
181		Mode					2	x								0	If relevant
182		External Interfaces					2	x								0	If relevant
183		Functional Requirements					2	x								0	If relevant
184				Functional Requirement			2	x								0	If relevant
185		Performance					2	x								0	If relevant
186		User Class					3	x								0	If relevant
187				Functional Requirement			3	x								0	If relevant
188		Classes/Objects					4	x								0	If relevant
189		Class/Object					4	x								0	If relevant
190		Attributes					4	x								0	If relevant
191				Attribute			4	x								0	If relevant
192				Functions			4	x								0	If relevant
193				Functional Requirement			4	x								0	If relevant
194				Messages			4	x								0	If relevant
195		System Features					5	x				x				2	Detailed Requirements*
196				System Feature			5	x								2	Detailed Requirements*
197							5	x					x			2	Detailed Requirements*
198				Introduction/Purpose&Priority Stimulus/Response Sequence			5	x					x			2	Scenario*
199				Associated Functional Requirements			5	x						x		2	Detailed Requirements*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
200						Functional Requirement	5	x					x			-2	Detailed Requirements*
201			Stimulus				6	x								0	If relevant
202					Functional Requirement		6	x								0	If relevant
203			Information Flows				7	x								0	If relevant
204				Data Flow Diagram			7	x								0	If relevant
205					Data Entities		7	x								0	If relevant
206					Pertinent Processes		7	x								0	If relevant
207					Topology		7	x								0	If relevant
208					Process Descriptions		7	x								0	If relevant
209					Process		7	x								0	If relevant
210						Input Data Entities	7	x								0	If relevant
211						Algorithms/Formulas of Process	7	x								0	If relevant
212						Affected Data Entities	7	x								0	If relevant
213						Data Construct Specifications	7	x								0	If relevant
214						Construct	7	x								0	If relevant
215						Record Type	7	x								0	If relevant
216						Constituent Fields	7	x								0	If relevant
217						Data Dictionary	7	x							x	0	Included in Data Requirements
218						User Class	8	x								0	If relevant
219						Feature	8	x								0	If relevant
220						Introduction/Purpose&Priority	8	x								0	If relevant
221						Stimulus/Response Sequence	8	x								0	If relevant
222						Associated Functional Requirements	8	x								0	If relevant
223						Performance Requirements		x	x	x	x	x	x	x	x	2	
224						Computer Resources		x	x	x	x	x	x	x	x	2	Appendix*
225						Response and Processing Times		x								1	
226						Precision						x	x	x	x	1	
227						Capacities (Normal, Peak Level, etc.)		x	x		x	x	x	x	x	1	
228						Data Requirements		x	x	x						2	
229						Database		x		x						2	
230						Entities					x					2	
231						Attributes					x					2	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
232			Relations								x			x	x	2	
233			Data Model								x			x	x	2	
234			Data Dictionary													2	
235			Data Element				x								x	2	Appendix*
236			Name				x									1	
237			Representation				x									2	
238			Units/Format				x									1	
239			Precision/Accuracy				x									1	
240			Range				x									1	
241			Design Constraints				x			x	x	x	x		x	2	
242			Standards Compliance				x	x				x		x		2	
243			Implementation Constraints							x			x			2	
244			Software System Attributes				x			x	x	x	x			2	
245			Reliability				x	x			x	x				2	
246			Fault Tolerance					x								1	
247			Recoverability					x								1	
248			Availability				x			x	x			x		2	
249			Security				x	x		x	x		x			2	
250			Confidentiality												x	2	
251			File Integrity Requirements											x		2	
252			Privacy Requirements							x						2	
253			Maintainability				x	x		x				x		2	
254			Analysability					x								1	
255			Changeability					x								1	
256			Stability					x								1	
257			Testability					x								1	
258			Supportability													2	
259			Portability				x					x		x		2	
260			Adaptability					x								1	
261			Installability					x								1	
262			Conformance					x								1	
263			Replaceability					x								1	
264			Usability					x							x	2	
265			Ease of Use				x								x	2	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
266				Understandability					x							2	
267				Learnability					x			x		x		2	
268				Operability					x							2	
269				Scaleability							x			x		2	
270		Other Nonfunctional Requirements						x					x			2	
271		Operating Environment								x						2	Appendix*
272				Computing Platform (HW, OS, SW)							x	x				2	
273				Concurrently Running Applications&Components									x			1	
274				Physical Environment										x		-1	If relevant
275				Documentation Requirements								x				2	
276				User Documentation						x		x	x			2	
277				Online User Documentation								x				2	
278				Printed User Manual								x				2	
279				Tutorials								x	x			2	
280				Installation Guides								x				2	
281				Configuration								x				2	
282				Read Me File								x				2	
283				Validation Criteria						x					x	-1	
284				Test Classes											x	-1	
285				Expected Responses											x	-1	
286				Performance Bounds											x	-1	
287				Miscellaneous Requirements												1	
288				Audit Requirements										x		-1	If relevant
289				Cultural Requirements										x		-1	If relevant
290				Political Requirements										x		-1	If relevant
291				Legal Requirements										x		-1	If relevant
292				Packaging Requirements						x						-1	If relevant
293				Labeling Requirements								x				-1	If relevant
294				Licensing Requirements								x				-1	If relevant
295				Installation Requirements								x				-1	If relevant
296				User Training Requirements										x		-1	If relevant
297				Invariants							x					-1	If relevant
298				Business Rules										x		-1	If relevant
299				Likely Changes							x					1	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
328			Analysis Metrics To Be Used												x	0	
329			Other Supplementary Information												x	1	
330		Glossary							x	x	x	x	x			2	
331		Index						x			x	x				1	
332	Number of topics addressed						110	34	52	57	76	74	84	49			

APPENDIX 2: CASE STUDY DESCRIPTIONS

This appendix contains the case study descriptions of the BaRE evaluation project. The project was divided into two phases, of which Phase 1 lasted from the end of June 2002 to the end of December 2002, and Phase 2 continued from there till the end of May 2003. The time before this project is referred to as Pre-BaRE time.

The introduction section 1.1 provides information about the project participants, software and RE assessments, and Finnish RD template. The three case study descriptions (Sections 1.2-1.4) have six subsections each: Background, Improvement Actions, Requirements Development Projects, Changes, Benefits, Costs, and Lessons Learned. The background covers company, project, key participants, problems, goals, software and RE practices, and process improvement issues. The improvement actions provide information on the actions taken in each company and how they covered different BaRE constructs. The project section summarizes the projects that used the new practices and describes the requirements documents developed in them when possible. The changes section reports the changes in the company practices using different assessment frameworks: the Lightweight REAIMS Top Ten, SPICE, infrastructure, and working practices. It reports also the achievement of the goals set for different phases, how the problem areas changed during the project, and the next actions planned in RE. The benefits and costs sections summarize the benefits the companies were able to identify during the project and, on the other hand, the costs that could be allocated to this improvement effort. In most cases this means the use of time in different RE and process improvement related activities. In the last section the lessons learned from this project are documented. The first things the participating companies found important are summarized and then questions interesting from the research point of view are discussed. These include the role of the ready-to-use concept and the BaRE method, and how the BaRE method compares with the high level design requirements. The last part of the section summarizes more general lessons learned from the project that were not anticipated in the planning phase.

1.1. Introduction

To be able to estimate the effect people had on the outcome of the project, some basic information about the participants was gathered. First of all the job titles, education, and experience in software development work of all the participants were recorded. The education is indicated with one of the three letters – B, B*, and M – where B stands for a Bachelor's degree, B* for a Bachelor level education in a Master's program¹, and M for a Master's degree. To be able to estimate the level of knowledge people had in RE also their competences were estimated. After considering People-CMM for the purpose (Curtis et al. 1997; Curtis et al. 2001) it was decided to follow a more lightweight approach developed from Vickers et al. (2002). This assessment scheme is not documented in detail in the public domain, but the available information served well the present purposes. The assessment was a self assessment and covered general competences, competence in process, and competence in RE techniques

¹ B* is due to the local university education system where a Master's degree can be achieved directly without a Bachelor's degree.

Appendix 2. Case Study Descriptions

and specification (see Appendix 3. Competence Evaluation Form). The competence in each area was estimated on five levels as shown in Table 1. Considering these levels it is clear that the BaRE method is targeted to help Novices to achieve a Trainee level by providing information for them and to support in the first few projects to move from the Trainee to Supervised practitioner level. Since changes in competences do not happen quickly, the competences were measured only once to see the level of knowledge the participating people had during the project.

Table 1. Competence levels for the competence assessment

Level of competence	Characterization
Novice	New to topic in question
Trainee	Knows theory
Supervised practitioner	Can do; supervision and pre-reviews are advantageous
Practitioner	Has done; works independently and participates in peer reviews
Expert	Proven ability; experienced in different domains and adaptation

The software and RE assessments were based on three frameworks: CMM (Software Engineering Institute 1995), REAIMS (Sommerville and Sawyer 1997), and SPICE (ISO/IEC TR 15504-1 1998). The CMM and REAIMS assessments were based on just interviewing the participating people and in the case REAIMS a lightweight adaptation of its top ten practices was used. The SPICE evaluation, on the other hand, was conducted by a trained assessor and followed the official guidelines for the assessment. Since one central aspect of the BaRE method is to improve the company RE infrastructure, an additional infrastructure list was developed and used to supplement the Lightweight REAIMS Top Ten assessment. When the availability of the suggested infrastructure was assessed, the following options were used: Not Available, Discrepant artifacts exist, Common artifact available in company, and Standard artifact available in company. Both ratings are comparable with the REAIMS assessment ratings except for the Standard use where only documented use of the practice was required; checking as a part of quality assurance procedure was not required. The use of the available infrastructure can also be assessed with the following options: Not Used, Partly Used, and Used. One project characteristic of interest in the case studies was the status of the project where the options followed general software development process: Specified, Designed, Implemented, Tested, Integrated, and Released. The RD baselines, on the other hand, were As Specified, As Designed, and As Released, which seemed to suffice well in these case studies.

The BaRE method v1.0 included requirements documentation templates, but all the templates were provided only in English. However, all the participating companies wanted to use Finnish documentation when possible and thus two companies did their own translations of the BaRE RD template. The third company hired a person who did not understand Finnish during the project and thus they were not interested in Finnish documentation any longer.

To be able to evaluate the Finnish documents, a BaRE translation of the RD template had to be developed. Consequently, the Finnish BaRE RD template was developed to resemble the English one as much as possible. In addition to investing a fair amount of own effort to find terms that best described the English concepts, also the translations companies had done own their own were considered.

Appendix 2. Case Study Descriptions

Company A developed first a Finnish RD template and then a Finnish RD based on that template. Company C, on the other hand, did not declare any company template yet but developed some templates on a project basis and used them on the discretion of the project managers. Thus for Company A both the template and the completed RD were compared, but for Company C only the completed RDs were compared. After the companies had provided their RDs, they were also given the Finnish BaRE RD template for further use.

1.2. Case A

Company A focused on implementing the BaRE method in Phase 1 and ran a pilot project in Phase 2. The pilot project focused primarily on getting familiar with the new practices so this case account concentrates on the implementation of the BaRE method in full scale in a large company.

1.2.1. Background

Company A is the IT department of a large metal industry corporation. This department was established in the early 70's when the first computers were bought in the company and it was currently responsible both for daily IT operations and software development in the company. During the present study the department had 14 employees of which 7 were developing software in-house. In addition to own development, the department was also responsible for purchasing software from external vendors both for internal use and to supplement other products the company produced. To keep this task manageable the department focused on administrative and business applications (ABA); the embedded software used in actual products was left for the product development department. The software development involved normally numerous other people from the business and product development departments representing customers and users of the software.

A **typical internal project** included two people and took about 20 months [MO] to complete with an additional 6 months for the requirements development phase. Consequently a typical effort for RE was about 12 person months [PM], and the respective effort for development was about 40 PM. External projects tended to have 2 or 3 people from the IT department and the same amount of people from the supplier. The project characteristics are summarized in Table 2; the figures are only trend-setting due to heavy variance between projects. The external

Table 2. Typical project characteristics

Project characteristics		Typical	
		Internal	External
Team size [people]	RE	2	6
	Development	2	6
Duration	RE	6 MO	6 MO
	Development	20 MO	20 MO
Effort	RE	12 PM	27 PM
	Development	40 PM	90 PM
Application Domain		ABA	ABA

Appendix 2. Case Study Descriptions

projects column includes both internal and external participants (3 and 3 people), while in the effort external people were estimated to manage with half of the effort internal people use on projects. This effort distribution had been observed in numerous previous projects.

The two **key participants** in the present project were an IT manager (A1) and a system analyst (A2). The IT manager was responsible for daily operations, software development, and data security, but participated also in some development projects. The system analyst got involved in this project since she was chosen to run the first project with BaRE. The IT manager had 20 years' experience and a Master's degree, while the system analyst had four years' experience and a Bachelor's degree. In competence evaluation they both selected Supervised Practitioner level for six areas and Practitioner level for ten areas (Table 3).

Table 3. Key people in the BaRE evaluation project

Person	Title	Education	Experience [years]	Competence level				
				N	T	S	P	E
A1	IT Manager	M	20	0	0	6	10	0
A2	System Analyst	B	4	0	0	6	10	0

The **main problem** in software development was that everything was a bit unclear. Prime examples of the situation included problems of locating documents after project completion and lack of requirements test cases. Problems with software project delivery schedules had also resulted in harsh critic from their steering groups. To improve the situation an analysis had been conducted and two specific problem areas identified: requirements engineering and project management. When seeking for help in these areas the IT manager contacted the local university and was informed about the BaRE project. Thus the situation in the beginning of the BaRE evaluation project is well characterized by a quote from the IT manager: "As of now we know that something must be done."

In the beginning of the present project a number of general improvement **goals** were identified. The most important ones were the following:

1. Increase system delivery reliability, i.e. more reliable estimates on development effort and duration were needed.
2. Increase project progress visibility.
3. Introduce a way to specify systems so that customers, users, and development organization would understand them in the same way.
4. Establish a common way to do RE.

From this list it was decided to focus only on the last one – establish a common way to do RE – in the BaRE evaluation project, since it fitted best with the whole method validation goal. Due to the fact that only pilot projects could be run during the evaluation project, it was clear that no conclusive answer to achieving this goal could be provided. Thus this project focused on goals refined from the overall goal for Phases 1 and 2 (Establish a common way to do RE) as shown in Table 4.

Table 4. Goals for the BaRE evaluation project

Phase	Goal
1	<p>Document templates are defined and available.</p> <p>The standard location for documents is defined.</p> <p>A requirements development process and techniques are defined.</p> <p>A change management process and procedures are defined.</p> <p>A requirements management tool is adopted.</p> <p>People know how to use all new RE artifacts.</p>
2	<p>Test the new practices with a pilot project.</p>

Company A had defined guidelines for **software development** already about ten years earlier, but in practice people interpreted these guidelines in different ways and projects were run differently. There were no named methods or techniques used to support software development, but informal reviews were conducted on a discriminatory basis. No special purpose tools were used for any of the software development tasks, but a help desk and bug reporting software were used in customer support. All the six people doing software development were called system analysts, while the other titles in the IT department were IT manager (one person), project manager (one person), and support person (two people). The system analysts' role was dual – in internal projects they appeared to work as software developers while in external projects they worked much more as application domain specialists backed up by actual business and application area expertise from respective departments. The analysts had also specialized to some extent in different areas in software development based on their personal interests.

The discussions on internal **software and requirements engineering practices** with the IT manager and the system analyst suggested that Company A was on initial level in CMM. For example, no practices had been documented, no special purpose tools were used even in configuration management, and project management had been identified as a key problem. This estimate was supported by the Lightweight REAIMS Top Ten assessment where seven practices were never applied, two were applied on the project manager's discretion, and only one practice was applied on a normal basis; no practice was on the highest level of standardized use. The requirements document development was mostly left for the project manager, but in some cases also other project members or software contractors took care of this task.

A need for software **process improvement** was evident in Company A. The company had participated in a general process improvement project four years earlier without much effect. More recently one project in another company of the group had used RUP in one of their projects in which the IT manager participated as an expert group member. One of the most important lessons from this project was the importance of proper specification phase; this specification project lasted 4 months, involved 10 external and 5 internal people, and resulted in a 374 page specification. The key issue with the process improvement effort was then how to get it started.

1.2.2. Improvement Actions

Company A took the BaRE method in effective use and addressed all the RE introduction issues in the way suggested in BaRE. They started by naming a RE coordinator, purchased a requirements management tool and tailored it to support their requirements development and management practices. At the same time the BaRE requirements document template was implemented in the RM tool. To ease the acceptance of all these changes, a wide-range training program was run among all the people involved in RE. This section provides a closer look at all these improvement actions focusing on the infrastructure that was developed for use in actual RE projects.

The system analyst A2 was named as a **RE coordinator**. Since she was planned to run the first project with the new practices, this nomination appeared justified, especially since she indicated interest in working as a RE coordinator also in the future. No other specialists were named during the present project although plans to nominate respective people in project management and testing were initiated.

The **RD template** required some modifications as it was to be created by the RM tool instead of text processing software. The main difference was that the single RD template in BaRE was split to six different documents that supported the working model of the tool better. This change was suggested by the tool consultant and he also implemented all the changes. Changes in the document headings, on the other hand, were few in numbers: 47 of the headings were identical, 2 were similar but used different wordings, 2 headings were omitted, and 6 headings were added. The differences were also marginal in practice since the omitted ones were introduction sections and the added ones presented more detailed topics in existing sections. The Finnish template, on the other hand, differed quite a bit from BaRE template: 12 of the headings were identical, 29 similar, 6 different, 4 omitted, and 5 added. The differences in the headings narrowed the scope of the heading in two cases and changed the meaning to a different one in four cases. For example, dictionary translations to one of the Finnish headings are “scaleability” (BaRE) and “expandability” (Company A). The added and omitted headings included similar ones as the English template. The Finnish template included an additional use case template with 21 headings, which is far more than even suggested in the BaRE Use Case Description appendix. Despite the differences in the document headings, the RD templates can be considered very similar. A final point lessening the differences between the Finnish templates is the fact that they both include the same English descriptions and examples for the contents.

In the end of Phase 2 a requirements document could not be printed as one document ready to be stabled and distributed. This functionality was not available in the selected RM tool configuration and it had not been considered important so far. In the same vein the template for RD was not yet polished – it had only a simple cover page with limited document information and table of contents, while the BaRE document template followed a comprehensive national standard on these issues. The limited interest in the outlook of the document followed from two facts: first, the document was mainly for internal purposes, and second, all the people needing the information had an on-line access to the actual requirements database.

Both the **requirements development and management practices** were defined in Phase 1 and they were both based on the BaRE practices. The requirements development process had nine

Appendix 2. Case Study Descriptions

steps that were directly copied from the BaRE process, but included additional tasks to support the effective use of the selected RM tool. The nine steps were also grouped into two phases – requirements definition phase and requirements under change control phase. The first phase included the first four steps from the BaRE process and they were to be repeated until the requirements could be considered ready. This phase was followed by the rest of the steps in the BaRE sequence. Since the BaRE documentation was used as the kernel for the process definition, the BaRE techniques formed the basis for the requirements development practices. The requirements management process was documented in a 26 page guide describing how the defects and change requests were to be managed. The process had the same three steps as BaRE but they were described in much more detail in the process guide. All the RM techniques suggested in the BaRE Guide were suggested in the process guide or implemented directly by the RM tool.

A solid **RM tool** formed the basis for new the approach to RE. Due to the long lifetime of the basic products of the company, the persistence of development information was found crucial and a well-established commercial database-based RM tool was considered necessary to satisfy these needs. Discussions with major RM tool vendors were started almost a year before the BaRE evaluation project started, and half way through Phase 1 of the present project the DOORS RM tool from Telelogic was selected. The selection was based on numerous contacts with the vendors and the general confidence these meetings developed on the supplier's capabilities to provide a solid solution for RM; no detailed assessments or comparisons were made. The needs for customization were discussed in meetings and a three-day workshop with a vendor consultant, who also made the changes in the tool. The overall goal was to get the tool in production use with company specific processes and templates to get real experiences from it in use. Thus a number of details were left to be considered later with experiences of the tool. Such details included e.g. printing of a single finalized document mentioned above.

The primary **tool for change management** within the development team was DOORS, while the customer requests were channeled through a HelpDesk software from the TJ Group. The customer support worked as the primary interface to users and handled general support calls as well as defect reports and change requests. Due to the importance of these contacts the HelpDesk software had already been used in support for four years. The new change management process did not change this customer interface, but clarified the management of change requests thereafter: the change requests were collected and reviewed when need arose, and requests requiring development attention were moved to the change management module of DOORS to be handled by the development team.

Serious investments in **training** and studying were made during the project. Training was a normal part of operations in Company A, but the RE improvement effort caused a serious peak in this activity. The time spent in studying was most noticeable with the IT manager and the System analyst, but practically everybody else participated in the arranged training sessions. Table 5 shows the various information sources that were used and how different people used them. In the table *RE Presentations* refers to the presentations given by the author of this thesis, and *RE Courses* refers to the commercial training about writing requirements and using DOORS. Telelogic provided both these courses and also the *Specialist Support/Consultation* that was needed to adapt BaRE to the needs of Company A. *Software Engineering Training* before the evaluation project had addressed RE, testing, and project management areas, while in Phase 1 it included only a software process improvement course arranged by tSoft. *Related*

Appendix 2. Case Study Descriptions

Literature was only ready by the IT manager who kept himself updated by reading software engineering-related literature all the time. However, in Phase 1 he familiarized himself with over ten books in related areas. In Phase 2 four people took the requirements writing course they missed in Phase 1 and everyone participated in a new project management course. In Company A no *Application Domain Training* was arranged in during the present project.

Table 5. Training and information sources utilized by different people during the different phases

Information Sources	Pre-BaRE		Phase 1			Phase 2		
	A1	Others	A1	A2	Others	A1	A2	Others
BaRE Guide			x	x				
BaRE Templates			x	x				
Practical RE in Short			x	x				
RE Presentations			x	x	x			
RE Courses			x	x	x			x
Specialist Support/Consultation	x	x	x	x	x			
Software Engineering Training	x	x	x	x	x	x	x	x
Application Domain Training								
Related Literature	x		x					

Company A did not run **other process improvement projects** simultaneously with the BaRE evaluation project. The project management improvement activities were running parallel to RE and a new effort in metrics area was initiated in Phase 2 to be able to measure the software engineering in the company. The RUP project ran in another company of the group got completed during the present project, and the number of establishments utilizing it increased steadily thereafter; this project was considered a success by all measures. However, the impact of these other efforts was deemed insignificant from the RE improvement point of view and BaRE was considered as the primary model for RE.

The improvement actions in Company A can be summarized by saying that a comprehensive RE method was adapted to company needs in Phase 1. Basic training was provided so that people would get the skills needed to utilize the developed infrastructure effectively in practice. These practices and skills were put in an actual test in the Phase 2 projects, which are considered next.

1.2.3. Requirements Development Projects

Company A used Phase 1 for establishing infrastructure and preparing personnel for the new way of working and did not run any projects with the new practices then. However, it was decided that one incomplete project (project A-P1 in this thesis) would be continued with the new practices in Phase 2. The new change management practices were planned to be tested with a new release of an existing software product in Phase 2. During the project it became

Appendix 2. Case Study Descriptions

apparent, though, that this was not realistic with the available resources and no action was taken in this area.

The key elements tested in the A-P1 project were the new RD template and the RM tool. The project was started with an old RD template and practices; in Phase 2 the document was converted to DOORS, and the missing sections in the document were patched. The RD was then reviewed and updated before it was accepted as ready for development. In the end of Phase 2 the project was implemented and testing had started, but the development and integration effort and duration had not yet been calculated. Thus these are presented as estimates in the project key characteristic summary in Table 6.

Table 6. Key characteristics of the project

Project characteristics		Project A-P1
Team size [people]	RE	4
	Development	4
	Integration	3
Duration	RE	5 MO
	Development	4 MO
	Integration	4 MO (e)
Effort	RE	8 PM
	Development	10 PM
	Integration	3 PM
Application Domain		ABA
Project Status		Implemented

The A-P1 project was different from a normal software development project. The project was initiated to move a complex database system from old hardware to new a one and to change the database version, tools, etc. at the same time. Thus an RD was needed to document all the issues that had to be taken care of in the project. The RD was considered crucial for the project, even though the technical details had to be addressed in much more detail in a technical specification. The different nature of the RD was well reflected in the document contents – only 25 of the company-specified 74 topics were addressed in the RD. Of these seven were headings for a group of subtopics without contents; for the system requirements there were 12 document specific subheadings that included detailed descriptions for 108 requirements. Altogether the document had 38 pages of requirements.

As the conclusion of the pilot project, the IT manager summarized that the A-P1 project utilizing the new requirements development practices was a success so far. The next thing on the project front was the change management that needed more effort to get it working.

1.2.4. Changes

The biggest change in Company A in the RE process improvement area was that in the end of Phase 2 they knew what it was all about. That is, they started to recognize the problems they were experiencing, reasons for the problems started to show, and ways to improve the situation became also evident. Thus in the end of Phase 2 they were involved in the actual ‘doing’ phase. In general software development did not experience other changes during the present project: both custom and COTS software were still developed and no other methods or tools were taken in use.

The **Lightweight REAIMS Top Ten** assessment result showed a dramatic increase from 4 points to 21 in Phase 1, as shown in Table 7. However, since this estimate is based only on one project and is not a real assessment result, another more straightforward assessment was done with the forms shown in Table 8 and Table 9. Since the infrastructure was introduced in Phase 1 and only a pilot project was run in Phase 2, the Lightweight REAIMS Top Ten point gains were the same in both the phases. The SPICE RE process capability assessment result in Company A was the following: “Process assessment was conducted on levels 1 and 2 and level 2 was achieved. Level 1 assessment covered performance of 8 base practices, and as most of these practices were fully achieved, the overall rating was that the level 1 performance attribute was fully achieved. Level 2 assessment covered 8 management practices both in performance and work product management attributes, and the level 2 was fully achieved” (Saastamoinen and Tukiainen 2003a, p. 18).

Table 8 shows well the Phase 1 focus in building **infrastructure** and the marginal changes in it in Phase 2. Most of the infrastructure was based on BaRE, but some adaptations to company needs were done. No training was defined, but appropriate training was obtained in the areas considered most important.

Table 7. Lightweight REAIMS Top Ten assessment point gains in different phases. The symbols mean the following: ‘-’ Never applied, ‘O’ Applied at the discretion of the project manager, ‘◐’ Normal use, and ‘●’ Standard use in company

Lightweight REAIMS Top Ten Guidelines	Pre-BaRE	Phase 1	Phase 2
Define practices for RM	○	●	●
Define a standard document structure	○	●	●
Define unique identifier for each requirement	-	●	●
Define standard templates for requirements description	-	●	●
Define validation checklists	-	●	●
Organize informal requirements reviews	◐	◐	◐
Use language simply, consistently and concisely	-	○	○
Make the document easy to change	-	○	○
Use checklists for requirements analysis	-	○	○
Plan for conflicts and conflict resolution	-	○	○
Total Point Gain	4	21	21

Appendix 2. Case Study Descriptions

Table 8. Infrastructure in different phases. The options were the following: Not available, Discrepant artifacts exist, Common artifact available in company, and Standard artifact available in company

Element	Pre-BaRE	Phase 1	Phase 2
Templates			
Requirements description	Discrepant	Standard	Standard
Requirements document	Discrepant	Standard	Standard
Change requests	Discrepant	Standard	Standard
Processes			
Requirements development	Not available	Standard	Standard
Change management	Discrepant	Standard	Standard
Review	Discrepant	Standard	Standard
Techniques			
Requirements development	Discrepant	Standard	Standard
Change management	Discrepant	Standard	Standard
Checklists			
Requirements engineering	Not available	Standard	Standard
Change management	Not available	Standard	Standard
Training material			
Requirements engineering	Not available	Standard	Standard
Software engineering	Not available	Not available	Discrepant
Application domain	Not available	Not available	Not available
Methods			
Requirements engineering	Not available	Standard	Standard
Software engineering	Not available	Not available	Not available
Tools			
Requirements management	Not available	Standard	Standard
Change management	Standard	Standard	Standard

The actual **use of the practices** is shown in Table 9. The Pre-BaRE and Phase 1 columns present estimates of the practice use in the company in general, while Project A-P1 in Phase 2 presents the practices as they were actually done in the project. That is, in the requirements development phase eight of the nine process steps were completed partly, and seven of the 18 techniques were used. In the requirements management activity no process was followed in the development phase, but six of the nine BaRE techniques were used.

The main **goal** for Phase 1 was the establishment of a solid infrastructure for RE, while Phase 2 was to run pilot projects with them. The goal of Phase 1 was achieved as planned, but the goals of Phase 2 were achieved only partially. The A-P1 project utilizing the new requirements development practices was considered a success in the end of the implementation phase. Table 10 summarizes these achievements.

Table 9. Use of the practices in the different phases

Element	Pre-BaRE	Phase 1	Phase 2 A-P1
Templates			
Requirements description	Partly	Partly	Used
Requirements document	Partly	Partly	Used
Change requests	Partly	Partly	Not
Processes			
Requirements development	Not	Not	Partly
Change management	Not	Not	Not
Review	Used	Used	Used
Techniques			
Requirements development	Partly	Partly	Partly
Change management	Not	Not	Partly
Checklists			
Requirements engineering	Not	Not	Not
Change management	Not	Not	Not
Training			
Requirements engineering	Not	Used	Partly
Software engineering	Partly	Not	Used
Application domain	Not	Not	Not
Methods			
Requirements engineering	Not	Not	Used
Software engineering	Not	Not	Not
Tools			
Requirements management	Not	Not	Used
Change management	Partly	Partly	Partly

In the end of Phase 2 the most urgent **problems** discussed in the beginning of the BaRE evaluation project were solved in the A-P1 project. The requirements documentation was located in the RM tool, and a network directory to other documentation was included in the RD, project tracking was satisfactory to the IT manager, and RE was completed in an orderly manner. The larger issues of testing and being able to adapt the process were also addressed to some extent: the system test cases were documented in the RM tool and a standard process that allowed adaptation to project specific needs appeared satisfactory to the IT manager. It was admitted that in the projects ran by the system analyst A2 such problems were fewer also in general, but still the A-P1 project was considered as an important forerunner in RE area. The company infrastructure in RE was estimated to be in a good shape and the next problem areas in RE were to get all projects utilize this infrastructure and raise the know-how of the other people to a level similar to that of A2. Outside RE, the main problem was managing the software development as a whole. For example, the change management was a problem, as prioritizing change requests is not possible unless they are documented and evaluated en bloc. As a solution to this problem a more systematic way to develop software was desired.

Table 10. Main achievements

Phase	Achievements
1	<p>Requirements document, requirements description, and change request templates adopted.</p> <p>Standard location for requirements documents defined as the RM tool.</p> <p>Requirements development practices defined: process, techniques, and tasks.</p> <p>Change management practices defined: process, techniques, and tasks.</p> <p>A requirements management tool adopted.</p> <p>People trained in RE.</p>
2	<p>A project swapped to the new practices in the end of requirements development phase and requirements completed with new practices; in the end of development phase the project appeared successful.</p>

The **next steps** in the RE area after Phase 2 were to get all the new RE infrastructure and working practices in full scale use in the company. One short project was expected to start shortly after the end of Phase 2, which meant that basically all RE projects in one year from the start of the BaRE evaluation project were to be run with BaRE.

1.2.5. Benefits

The benefits of the BaRE evaluation project can be summarized as two areas: introduction of standard infrastructure for RE in the company, and positive experiences with the pilot project A-P1. The existence of standard company infrastructure made it possible to expect its use in every project, which, on the other hand, was expected to make project tracking easier and improve the quality of projects.

In the A-P1 project these gains were recognized in an intuitive level both by the IT manager and the system analyst. As concrete outcomes of the good RE practices in the project they estimated that the project was well in schedule in the end of the implementation phase and the project tasks had experienced only about 10% changes in the design and implementation phases. Both these outcomes were encouraging, and supported the idea that RE was moving to the right direction now.

1.2.6. Costs

To get an idea how much all these changes cost, time, effort, and money spent to achieve them were considered. The first contacts with the tool vendors had already been established in the previous fall, but the final decision to purchase a tool was made in June after the initial state interview for the present project, and the tool selection was made in October. The effort in different activities is summarized in Table 11 for the IT manager *A1*, the System analyst *A2*, and other people in the company (*A-Others*). The *tSoft* column reports the time used for process improvement activities arranged by tSoft, but excluding the time used for RE-specific activities. The *RE Info* column includes different presentations of RE by the author of this thesis; this column includes also the presentations given as part of other tSoft seminars. The

Appendix 2. Case Study Descriptions

Meetings column refers to the meetings with the author of this thesis. The *Training* column includes all the training arranged in the company. The *Study* column and *RE Dev* columns refer to the time used in RE-related studying and development, like participating in a workshop and adapting the method and templates. The *Total* column sums up the use of time for each person and the company, while the *Totals* row shows the total time used for different activities.

Table 11. Time use in hours for RE and process improvement-related activities

Person	tSoft	RE Info	Meetings	Training	Study	RE Dev	Total
A1	10	3	16	49	140	162	380
A2	4	3	15	64	15	74	175
A-Others	7	13	16	706	0	127	869
Totals [h]	21	19	47	819	155	363	1424

In Company A most of the RE actions were completed in Phase 1, and Phase 2 contained only two person days of training and some meetings in RE. A project management course, on the other hand, added altogether 600 hours to the training time in Phase 2. The tSoft activities in Phase 1 included four working days of training, and in Phase 2 the SPICE assessment took about two working days. It was noted that the amount of training in this period was exceptional, and the normal amount of training was around two to four person days a year.

The only direct costs in Phase 1 were the purchase of the tool and training. To start with, seven licenses were bought, and the training included about 20 days of training and consultation covering process modeling, requirements writing, tool use and administration, as well as on site installation and adaptation of the software in a ready-to-use condition. The costs for the licenses and training were about equal and the time used by company employees on the project also resulted in about similar calculatory costs. Thus the total costs of the improvement effort for the company were formed by three equally big expenses – purchase of licenses, purchase of training and consultation, and company time used in the improvement effort.

Since the RE improvement effort involved major investments, calculations were made to estimate the return on the investments. The calculations were done by the IT manager and they were based on a 10% reduction in effort resulting from the introduction of the RM tool; the possible affect of the BaRE method to support the whole RE was not considered in them. The calculations suggested that the tool investment would pay itself back in two years.

1.2.7. Lessons Learned

The most important lessons to Company A from the adoption of the BaRE method were an increase in general knowledge about RE and establishment of own standard RE infrastructure. The increased knowledge with the splitting of RE into smaller units led to a more structured understanding of RE and to a better touch with the work that must be done. This, on the other hand, made the requirements development task much smoother than the previous approach of focusing on just writing down requirements. The experiences of the increased investment in the requirements development phase appeared also to ease the implementation phase. Consequently, the idea of doing requirements development as an own project was considered interesting, since it would allow proper planning of the implementation project based on known

Appendix 2. Case Study Descriptions

requirements. The working practices themselves did not change much during the project, but the previous ad hoc approach was replaced by a more systematic way of working. For the IT manager the two most important lessons were first realizing that RE documentation is really a valuable asset to the company, and second, an establishment of a roadmap to RE practices in the company: what the current situation with respect to the RE field was in general, where they should head for next, and what would need to be done to get there. The actual goal of improved software project performance did not materialize yet, since no projects were completed utilizing the new standard approach to RE in the company.

In the beginning of Phase 1, questions about the **ready-to-use concept** were asked of the key participants. At that time the most important properties of BaRE leading to the decision to adopt it on an as-is basis were the following:

- It provides a simple and understandable model on how to introduce RE.
- It is a thought-through model so it is probably best to try it and modify it if need arises.
- It appears very logical in the sense that no part in it appears unrealistic.
- Absolutely better approach than anything we are currently using.
- It is on the right level for our company needs considering the time needed to learn and adopt it. RUP takes so much time to learn that adopting it is impossible for us.

The interest in ready-to-use methods did not diminish during the project. In the end of Phase 1 the IT manager expressed interest in a ready-to-use software engineering method and in the end of Phase 2 the key interest was in such a testing method – if such were available. Considering some of their suppliers, the IT manager estimated that many of them would benefit from ready-to-use methods. However, he had not seen any other ready-to-use method before the BaRE method. As the key benefit of the ready-to-use concept the IT manager considered the limited need of own effort required to establish initial practices that can be taken in productive use in a company.

The BaRE method was considered as a ready-to-use method. However, on the basis of the experiences the IT manager found the adaptation to organization specific circumstances essential. In Company A the need for adaptation was evident for two reasons: first, many people had experience of different kinds of RE artifacts and practices, and consequently, they wanted to see some of these included in the company standard approach, and second, integration with a commercial RM tool required some actions. From the beginning it was clear that the BaRE method was to be used as the company RE method and no real alternatives for it were seen, although RE practices in five other companies were considered. The next best alternative was RUP, but it would have required extensive adaptation especially in comparison with the BaRE method. Thus the reason to choose the BaRE method as the company RE method was its readiness-to-use.

The estimates of the **fit-for-need** of BaRE in Company A were encouraging in the beginning of Phase 1, and in the end of Phase 1 this conception had only strengthened. Both the IT manager and the System analyst felt that the better one understood BaRE the better it looked. A few changes in the original BaRE were suggested and implemented by the RM tool consultant; most of the changes were cosmetic in nature, while some were required for an effective use of

Appendix 2. Case Study Descriptions

the tool. The resulting modified BaRE method was considered suitable as a standard way to run RE projects in the company. This property was essential for Company A, since it eased the project management by increasing project predictability, which again was among the biggest concerns in the company. The fit-for-need was among the prime factors for selecting BaRE in the first place, as it appeared to consider all the important areas in the application domain of Company A which was administrative and business applications. In the end of Phase 2 BaRE was still considered fit-for-need and clearly a better way to operate than the previously used ones.

As a whole, the BaRE method was found fairly **easy-to-use** in the end of Phase 1. A key property in this respect was considered the ease of adoption, as it was felt that it should be possible to learn the method along with the daily work without excessive amount of dedicated effort. BaRE fulfilled this requirement well and it was considered an understandable, compact, and simple method suggesting a useful way of working. The adoption in Company A required only a relatively small effort and the approach felt even more logical after having worked with it in practice. Two things explicitly mentioned to ease the adoption were previously used practices and reduction of RD templates to only one. That is, BaRE did not actually introduce many new techniques in Company A, but only gave names to the techniques that were used before and suggested a systematic way to do them. The change to a single RD template, on the other hand, resulted in a more unified picture of the RD as such and reduced competing definitions between different requirements documents and their parts. The systematic approach suggested in BaRE was found important for a basic method, as it made the adoption a straightforward process. From this point of view it was also important that the process made concrete suggestions on what to do in each step. The problems in the adoption phase were mainly caused by new topics in the RD template, as the earlier templates did not consider the requirements from such a wide perspective as the BaRE template. Many of these issues were resolved by the examples provided in the template, but a walkthrough of the template with concrete discussion what each topic meant in Company A would probably have eased the adoption.

The responses to the ease of adoption were much more realistic in the end of Phase 2. At this time the need to really study the method in the beginning was emphasized and the “just take it in use” approach was deemed unrealistic. However, after the initial acquaintance, BaRE was found easy to use. The provided training material was found sufficient, understandable, and logical; as a matter of a fact additional documentation was judged as an extra burden and was not wanted. The system analyst A2 did not use the method documentation in Phase 2, but she expected to use it again once the next project started.

Company A put the **comprehensiveness** of BaRE to test by addressing all the five key constructs for a ready-to-use method in RE, and went even beyond them. The five areas were, however, considered to be sufficient for introducing RE in a company, and the other issues could be considered as adaptation to company specific needs. The only RE practice implemented by Company A that was not suggested as a basic practice in BaRE was development of requirements test cases in the requirements development phase. This practice is, however, included as a part of the Detailed Requirements appendix so the scope of BaRE was well within the needs of Company A. The more general needs in Company A were related to the large projects they had – support for project management, metrics, and architecture were mentioned in some of the discussions. The general assessment was, however, that BaRE

Appendix 2. Case Study Descriptions

included all the essential elements and it was comprehensive enough for the needs of Company A.

The basic approach for the improvement effort was to select a commercial RM tool and use the BaRE method for all the other **key constructs**. The limited experiences from the practice included an observation that a RD was crucial for the projects run. The suggested BaRE practices for requirements development and management were adopted as the company standard in practice as they were for the infrastructure part. The actual working practices, on the other hand, changed only little in the pilot project. No standard training was defined in the company, but appropriate training was obtained in the areas considered the most important, and a fair amount of time was spent on self-study. From the suggested techniques checklists seemed the only item that was not even tested in practice.

As more **general lessons** from the BaRE evaluation project the IT manager concluded that the change from old practices to new ones was an agonizing one. It was estimated that this followed from the large size of the company and the inertia numerous stakeholders created. This problem did not come as a big surprise, as in the beginning of the improvement effort the author of this thesis was called to give a general introduction to the BaRE method and RE in general. The system analyst, on the other hand, found the requirements document very useful for the project A-P1 and estimated that they could not have actually started the project without such a good RD. The BaRE RD template was, however, very different from the ones she was used to, which made the first use of the template cumbersome. Thus the first project was considered as a practical exercise to get familiarized with the new method, but it was expected to ease the next project with full scale use of the new practices considerably.

Both key participants were skeptical about the progress that would have happened in case only the BaRE documentation had been available. Company A did not receive direct support for their effort from the author of this thesis, but the visits made the people study the documents before meetings and think about RE. As a whole the author of this thesis acted as a champion (Humphrey 1989, p. 31) for the method; if someone else had taken this role the outcome could have been comparable with what was achieved now. However, in Company A an external champion was considered very important for the success of this project.

As a whole the BaRE method was considered helpful in introducing the new RE practices in Company A. This estimate was supported by a promise to tell about the BaRE method to other people struggling with immature RE practices.

1.3. Case B

Company B needed immediate support in their daily software development, and consequently they took a phased approach to method adoption. The first priority was solving change management problems, which were addressed in Phase 1. Phase 2, on the other hand, focused on requirements development, as planning of a new major software version was initiated then. This case study describes the extremely agile software process improvement efforts in a small company living on a single COTS product.

1.3.1. Background

Company B was a five-year-old company focusing on a single system administration tool. The “single product” refers here to a single source code, since over half of the products sold were actually configured to customer preferences with a product configurator. Lack of customer specific versions was also reflected in the ratio of software developers and other personnel, as only three of the twelve employees were involved in software development.

A **typical project** for Company B was a new release that was completed every three to four months (Table 12). In this typical working mode, requirements were collected and managed continuously besides development with an effort of about one person month for each release. The software development staff consisted of three people of whom only one did not have other major tasks outside software development. Each release required a total effort of about seven person months.

Table 12. Typical project characteristics

Project characteristics		Typical
Team size [people]	RE	1
	Development	3
Duration	RE	Continuous
	Development	3-4 MO
Effort	RE	1 PM
	Development	7 PM
Application Domain		System Admin

In Phase 1 there were two **key participants**: a software manager (B1) and a software engineer (B2). The software engineer had been hired to the company just a month before the beginning of the present evaluation project and she continued to study at the local university towards her Master’s degree parallel to working. Her job was to support the software manager in software development, as he was doing both the management and coding parts of the development; the third person in the software development staff was a test specialist focusing solely on testing. The software manager had a Bachelor’s degree and 13 years of experience, while the software engineer had no actual working experience and she had done the studies required for a Bachelor’s degree (Table 13).

Table 13. Key people in the BaRE evaluation project

Person	Title	Education	Experience [years]	Competence level				
				N	T	S	P	E
B1	SW Manager	B	13	0	0	5	9	2
B2	SW Engineer	B*	0	4	11	1	0	0
B3	Senior SW Engineer	B	7	1	9	6	0	0

Appendix 2. Case Study Descriptions

Moving from Phase 1 to 2 happened simultaneously with major changes in the personnel. The software manager moved on in his career to another company and soon after that the software engineer decided that it was time to complete her studies, and she also resigned from the company. A new senior software engineer (B3) was hired to manage the software development, while another person was hired to concentrate on software development. The senior software engineer had seven years of experience and a Bachelor's degree (Table 13).

The **RE-related problems** were originally summed up as two basic problems: first, documents were not readily available when needed, and second, there had been communication problems between sales and software development people. Later on it was found out that software development knowledge had been limited to one person and documenting the knowledge was important for two reasons: first to be able to operate efficiently even if that key person was not available, and second, to qualify the company as a training job location for the local university students in computer science. During Phase 1 it became also evident that immature software development processes caused distraction to the developers to the point that something had to be done about it.

As the **goals** for Phase 1 in the BaRE evaluation project only two things were selected: first, collecting the dispersed documents in one place in a common format, and second, defining requirements management practices. A requirements development process (practices) and a requirements document template were also considered important, but a realistic view on the possibilities to invest in process improvement lead to assign them a low priority in Phase 1. Since these issues did not get addressed in Phase 1, they were considered the top priority in Phase 2 (Table 14).

Table 14. Goals for the BaRE evaluation project

Phase	Goal
1	Document existing requirements in a common format and place them in one data store. Define change management practices.
2	Adopt a requirements document template. Adopt requirements development practices.

The BaRE evaluation project was the first real **process improvement** effort in Company B, which was quite apparent, looking at their **software development and RE practices**. The only tool to support software development was the MS-Project – no special purpose tools were used for RE, design, testing, or configuration management. In the same vein no named methods were used and tester was the only specialized role in software development. The RE practices followed the general trend in the company, and only two practices had been established as a normal way of working: requirements were documented in a list format and changes were discussed in development group meetings before work was undertaken. Even these practices appeared informal by the REAIMS guidelines and all the top ten practices had to be considered as never applied, resulting in zero points in the assessment. This was in line with the informal CMM estimate suggesting an initial level company. The managing director for the company commented their practices as follows: “Our software development will not continue like this.”

1.3.2. Improvement Actions

The lack of resources to process improvement was evident from the beginning of the present project and thus the efforts were started with the most urgent problem – change management. Within change management there were two sub problems that needed to be addressed: documenting existing requirements and getting change requests under control. The different kinds of problems were attacked in different ways, first the individual requirements were just collected and written in an MS-Word document, and second, an RM tool was considered and taken in use to manage the change requests. Due to the small number of people involved in software development it was clear that the improvement work had to be done by the newly hired software engineer. All the people in the company, however, were provided a short introduction to RE to increase general knowledge about it.

The software engineer B2 was named as the **RE coordinator** half way through Phase 1. Due to her resignation in the end of Phase 1, the senior software engineer B3 took over the RE coordination role in the company in the beginning of Phase 2. No other specialist roles were assigned during the project.

The senior software engineer familiarized himself with the **RD template** in Phase 2 and started using it after he had considered developing an RD template of his own. The reason for considering an own template was due to perceived problems in using the BaRE template. However, these problems were caused by unawareness of the examples included in the BaRE template and after the template was discussed with the author of this thesis, it was taken in use with only one change. This change was an addition of a column in every table that showed the status of each requirement, so that all the requirements could be included in one document and reviewed individually.

The **requirements development practices** were considered quickly in the familiarization phase and were not needed much after that. The process and practices made sense and thus they were found as valuable support for learning systematic RE practices.

The biggest job was getting all the product-related information in one document. The final version of this technical specification had 30 pages, including a general description of the software and its components with detailed guidelines for change management and coding. Since all this information had previously been in discrepant notes and in the head of the software manager, this action created a solid basis for future development efforts.

The change request problem was attacked with an introduction of an **RM tool**. No studies were done between different tools but the tool presented in a tSoft seminar, (*sfrm* 2002)², was installed for a closer look, and as it provided clear improvement to the previous situation, it was taken in use. With some experience of the tool it was established as a standard solution to change management in the company.

² *sfrm* is a requirements management tool but the anticipated process for requirements development appeared to suit also change requests in required measures.

Appendix 2. Case Study Descriptions

The change management tool was supported by an **RM process** and a document describing how change management is conducted in the company. With the positive experiences from the introduction of the change management process another process to implement customer specific configurations was set up and taken in use. The configuration did not require changes in the code, but the work was done by the developers, and consequently, it had disrupted the development work before. Even though the new processes caused some initial resistance, both the development and sales people found them invaluable as a whole.

Training people was mostly left for self-study. The software engineer familiarized herself with all the provided material and the manager with the most essential parts of it. Due to her ongoing studies, the software engineer was also exposed to other RE and software engineering literature. The author of this thesis held one general presentation on RE to seven people who were connected to software development and requirements, while the other software engineering training was limited to tSoft events. The senior software engineer relied on his self-studies and the support he got from the author of this thesis. As shown in Table 15, only the software engineer familiarized herself with some related literature and before the project no training was acquired or training material was studied in these areas.

Table 15. Training and information sources utilized by different people during the different phases

Information Sources	Phase 1			Phase 2	
	B1	B2	B-Others	B3	B-Others
BaRE Guide		x		x	
BaRE Templates	x	x		x	
Practical RE in Short		x		x	
RE Presentations	x	x	x		
RE Courses					
Specialist Support/Consultation				x	
Software Engineering Training	x	x	x		
Application Domain Training					
Related Literature		x			

The only **other process improvement project** during the BaRE evaluation project was tSoft, which got the whole thing started in the company. Consequently, there were three primary information sources for the improvement efforts in Phase 1: tSoft and its SPICE training, the BaRE project and material, and the university courses taken by the software engineer. However, when the software engineer developed the first change management process, she did not use any material to assist her in this job, but devised the process from common sense and tested it with the experiences she had just gained in software development without processes. In Phase 2, due to the changes in personnel, only the BaRE material was used in improving the RE. The smaller improvement efforts in the company included five main actions: adopt a coding standard for naming variables etc., develop and implement a manual project table for project planning and scheduling, start using the MS-Project for project planning, start using a configuration management tool, and use the university usability laboratory to test the new

Appendix 2. Case Study Descriptions

software version before its release. From these the first three actions were already effective in the end of Phase 2, and the next two were also planned and scheduled to be implemented.

Company B's improvement actions appeared to happen overnight without much a priori planning and resulted in immediate benefits in daily work. This approach was evident both in Phase 1 with the change management practices and in Phase 2 with the requirements development practices. The BaRE method was stated to have speeded up the development and also enhanced the scope of the effort, since no such material was available earlier. The project utilizing the new requirements development practices is described next.

1.3.3. Requirements Development Projects

In Phase 1 no major projects were completed, but continuous development of the basic product with three versions and altogether ten releases was done. The technical specification for the product was developed and the change management practices with tool support were taken in use in the company. These activities were completed about half way through the first phase and the planning work for the next major release of the software was initiated. However, this activity halted due to the changes in personnel, and it only got on again after the senior software engineer was introduced in the company and could start effective work on specifying the software.

In Phase 2 the new major release was specified (the present project B-P1) and the use of the adopted RM tool and practices was continued. The RM tool was used both to record change requests for the current software version and to suggest new requirements to the new version. The B-P1 project, on the other hand, based its specification on the BaRE RD template and utilized also some of the BaRE development techniques. The project characteristics are shown in Table 16.

Table 16. Key characteristics of the project

Project characteristics		B-P1
Team size [people]	RE	1
	Development	3 (e)
Duration	RE	3.5 MO
	Development	5 MO (e)
Effort	RE	1 PM
	Development	11 PM (e)
Application Domain		System Admin
Project Status		Specified

In the end of Phase 2 the B-P1 development was started with a requirements document. The actual document was supplemented with three appendices on detailed requirements, interface specification, and use case descriptions, of which the first one was reviewed and accepted, while the other two had been started. The need for a glossary was also evident and developing it was waiting for a suitable moment. The developed RD was very similar with the BaRE template, and only some general properties and document headings had been slightly adapted

Appendix 2. Case Study Descriptions

to be used in a COTS project (Table 17). The Detailed Requirements appendix described 48 requirements in detail on 48 pages, while the Use Case Description appendix covered 10 use cases on 10 pages. The *Group header, empty* -count in Table 17 is provided only to be able to match the numbers in the table and it is not considered otherwise meaningful.

Table 17. Properties of the produced requirements document

Requirements Documents		B-P1
Available versions		B1
Page count contents/total		12/17
Appendix count		3
General	Identical	3
Properties	Similar	4
	Partly similar	2
	Different	0
	Added	0
	Omitted	0
	Count	9
Document	Identical	48
Headings	Similar	0
	Different	0
	Added	0
	Use case name	0
	Omitted	3
	Count	48
Contents for Headings	Group header, empty	8
	Not available	6
	Available	34
	Added	0
	Extended	0
	Modified	0
	Removed	0
	Count	40

1.3.4. Changes

The changes in Company B appeared to take place overnight. In the meetings things were discussed and in a follow-up meeting it was just reported that the RM tool had been taken in use, change management practices had been documented and adopted, and a requirements document had been written and reviewed. Before actions, the next issue needing attention had always been well recognized and it appeared that after that, at some point of time, the issue was just addressed and solved. The solutions were not any grand ones requiring extensive efforts but working solutions that really could be implemented in reasonable time frames. Otherwise

Appendix 2. Case Study Descriptions

the software development in the company did not change much: two people left and two full time and one part time developer started, development was still focused on a single product and no new methods were adopted. However, the RM tool was taken in use in Phase 1, and in Phase 2 the tool to support project management was adopted. In the specialization no actual changes happened even though the roles appeared now fairly clear: the senior software engineer was responsible for software development in general including requirements, one person worked as a fulltime software developer supported by a part time developer, and a fourth person was a fulltime tester.

The **Lightweight REAIMS Top Ten** assessment reflects well the completed changes in the company practices, since the point gain increased from zero points to 18 (Table 18). It is interesting to notice that even though Company B perceived the improvements in change management as very important for the company's RE operations, only one question in the REAIMS Top Ten questions addressed this area and, consequently, the total point gain after Phase 1 was only 3. The senior software engineer completed the personal competence evaluation form after Phase 2 to see whether anything had changed, and his capabilities had changed a lot. His competence in different areas had shifted one step higher in most of the areas and on average he was now on Supervised Practitioner level when the average five months earlier was on the Trainee level. Finally, the SPICE RE process capability assessment result in Company B was the following: "Process assessment was conducted on levels 1 and 2 and level 2 was achieved. The company had clearly defined a requirements management process and the level 1 performance attribute was fully achieved. Also level 2 management practices were achieved largely both in performance and work product management attributes. The company used the BaRE method to elicit and specify requirements and managed them with the *sfrm* software. These tools made it possible to achieve an established and predictable way of working and maintain requirements documents" (Saastamoinen and Tukiainen 2003b, p. 16).

The **infrastructure** in Company B changed quite a bit (Table 19). The RD template, as well as requirement templates were taken in use, and requirements development practices were defined to support the development process. A change management process was developed and implemented with a tool and a change request template. A review process was brought in by the senior software engineer, but it was not yet documented in the company. All the processes were fairly similar with the BaRE processes.

The actual **use of the practices** followed fairly well the defined practices (Table 20). Only checklists and training did not really get exercised in the company, but otherwise the BaRE method was utilized well. The introduction of the change management process appeared like the biggest change in the company, but even it started working well after initial inertia. The RM tool was used both to record change requests to the current software version and to suggest new requirements for the project B-P1.

Appendix 2. Case Study Descriptions

Table 18. Lightweight REAIMS Top Ten assessment point gains in different phases. The symbols mean the following: ‘-’ Never applied, ‘O’ Applied at the discretion of the project manager, ‘●’ Normal use, and ‘●’ Standard use in company

Lightweight REAIMS Top Ten Guidelines	Pre-BaRE	Phase 1	Phase 2
Define practices for RM	-	●	●
Define a standard document structure	-	-	●
Define unique identifier for each requirement	-	-	●
Define standard templates for requirements description	-	-	●
Define validation checklists	-	-	●
Organize informal requirements reviews	-	-	○
Use language simply, consistently and concisely	-	-	○
Make the document easy to change	-	-	○
Use checklists for requirements analysis	-	-	-
Plan for conflicts and conflict resolution	-	-	-
Total Point Gain	0	3	18

To get quick support for daily work, the **goals** for Phase 1 focused on change management, while in Phase 2 the requirements development practices were addressed. Thus change management practices were defined first and a requirements management tool was adopted to support them. At the same time the work to document existing requirements and development guidelines in one technical specification was started. After the change management process had been successfully taken in use, another process was defined to support customer specific configurations. In Phase 2 the BaRE RD template and requirements development practices were taken in use to support the requirements development for the new software version. Thus a comprehensive infrastructure for RE was established during the BaRE evaluation project and the goals for the project were achieved as planned (Table 21).

The most important **problems** in the beginning of the BaRE evaluation project were poor documentation as well as communication problems between the sales and the development people. The documentation problems followed from the lack of documentation, and when dedicated effort was conducted to develop them, this problem was solved. The most important aspect of the solution was the introduction of a standard tool to manage change requests. The lack of knowledge sharing was put to test when the two people left the software development, but it seemed that this change did not cause serious problems in the company. Getting the change management process in place with tool support was also reported to have resulted in clearly better communication between the sales and the development soon after their implementation. The systematic processes resulted also in fewer interruptions in the software development, since the change requests and the customer specific configuration requests followed standard processes. Finally the documentation on software processes experienced a serious improvement during the present project and in the end of Phase 2 no actual problem areas in RE could be pointed out.

Appendix 2. Case Study Descriptions

Table 19. Infrastructure in different phases. The options were the following: Not available, Discrepant artifacts exist, Common artifact available in company, and Standard artifact available in company

Element	Pre-BaRE	Phase 1	Phase 2
Templates			
Requirements description	Not available	Not available	Standard
Requirements document	Not available	Not available	Standard
Change requests	Not available	Standard	Standard
Processes			
Requirements development	Not available	Not available	Standard
Change management	Not available	Standard	Standard
Review	Not available	Not available	Common
Techniques			
Requirements development	Not available	Not available	Standard
Change management	Not available	Standard	Standard
Checklists			
Requirements engineering	Not available	Not available	Standard
Change management	Not available	Standard	Standard
Training material			
Requirements engineering	Not available	Standard	Standard
Software engineering	Not available	Not available	Not available
Application domain	Not available	Not available	Not available
Methods			
Requirements engineering	Not available	Not available	Standard
Software engineering	Not available	Not available	Not available
Tools			
Requirements management	Not available	Standard	Standard
Change management	Not available	Standard	Standard

In the end of Phase 2 the plans for **next steps** in the RE area included utilizing the existing RE infrastructure to a fuller extent and deepening the understanding of it. No further development in the RE area was planned and all the effort was now allocated to implementing the project B-P1.

Appendix 2. Case Study Descriptions

Table 20. Use of the practices in the different phases

Element	Pre-BaRE	Phase 1	Phase 2 B-P1
Templates			
Requirements description	Not	Not	Used
Requirements document	Not	Not	Used
Change requests	Not	Used	Used
Processes			
Requirements development	Not	Not	Partly
Change management	Not	Used	Used
Review	Not	Not	Used
Techniques			
Requirements development	Not	Not	Partly
Change management	Not	Used	Used
Checklists			
Requirements engineering	Not	Not	Not
Change management	Not	Not	Not
Training			
Requirements engineering	Not	Partly	Used
Software engineering	Not	Not	Not
Application domain	Not	Not	Not
Methods			
Requirements engineering	Not	Not	Used
Software engineering	Not	Not	Not
Tools			
Requirements management	Not	Used	Used
Change management	Not	Used	Used

Table 21. Main achievements

Phase	Achievements
1	Existing requirements documented in a technical specification. Change management practices defined and adopted. A requirements management tool adopted to support the change management. Control over changes established.
2	Requirements document template adapted and adopted. Requirements development practices adopted.

1.3.5. Benefits

As the most important benefit from the BaRE evaluation project was considered the fact that now the process improvement work had been started. In Phase 1 the more concrete benefits included introduction of systematic practices to replace the previously chaotic ones, and especially bringing change under control. Introduction of standard processes and a tool to support them had resulted in a standard electronic location and format for change requests. Thus the risk of losing them could easily be eliminated and, on the other hand, allowed basing development efforts on documented requests, not invented needs or rumours. The standard processes were also reported to result both in reduced interruptions in the development work and better communication between the sales and the development.

In Phase 2 the most important outcome was that the requirements had been discussed with different stakeholders and they had actually been documented and accepted in the company. The discussions on requirements had been conducted in reviews in the company for the first time now, and the first review resulted in 23 changes. This practice was actually brought in by the senior software engineer, but the review was considered useful and it seemed that the practice would be continued in the future. The benefit of having documented requirements had already made it possible to reject similar requirements from the outset, resulting in reduced effort spent on them. As a whole this improvement effort had created a comprehensive RE infrastructure for Company B, which provided a solid basis for future work with requirements and changes.

1.3.6. Costs

Company B's investments in the improvement effort were very reasonable. Namely, by selecting a freeware RM tool and relying on self-study Company B did not use any direct money in the improvement efforts.

The effort spent on the improvement seems fairly small since the total company investment was less than five person weeks. This does not include the effort spent on developing the technical specification, which was estimated to have taken about six person weeks. The improvements in requirements management area were implemented in about three months and even the documentation for it got completed within another three months. The calendar time spent in the requirements development area was considerably longer, nine months from the first concrete discussions, but in practice the BaRE documentation was somehow actively considered for about two months before the first requirements document was developed. The actual time the senior software engineer spent on studying the BaRE was less than one person

Table 22. Time usage in hours for RE and process improvement related activities

Person	tSoft	RE Info	Meetings	Training	Study	RE Dev	Total
B1	1	1	10	15	3	0	30
B2	4	2	5	0	21	32	64
B3	7	0	9	0	21	14	51
B-Others	14	9	7	0	0	0	30
Totals [h]	26	12	31	15	45	46	175

Appendix 2. Case Study Descriptions

week and the adaptation work took only half a day. Table 22 summarizes the use of time in Company B; people B1 and B2 participated in the project only in Phase 1 and person B3 only in Phase 2, while the B-Others row covers both the phases.

1.3.7. Lessons Learned

Company B did not end up with any grand insights as a result of the BaRE method adoption, but many practical things about RE were learnt. This outcome was natural, as the improvement actions were done as small units that solved practical problems in daily work. Another important property of the adoption process was that the personnel changed totally half way through the project. Thus three people got exposed to the BaRE method and they all seemed to learn the basics of it quickly. The fact that the senior software engineer wanted to complete the competence assessment both before he had any knowledge about the BaRE method and after he had completed his first RD with it provided some evidence from this learning experience – on the average he raised his competence level from Trainee to Supervised practitioner level.

The possibilities to validate the **ready-to-use concept** in Company B were limited in the beginning of the project since they only wanted to adopt some parts of the method and omitted an overall assessment of it. However, after the change management started to work and the interest shifted to the requirements development we returned to the readiness-to-use. At that point BaRE was considered as the starting point with which they wanted to get the requirements developed and only after getting experiences from BaRE it was found worthwhile to look at other approaches and possibly modify the method. The reason for postponing changes to a later point in time was a lack of time to do them. The software manager stated that the development of their system administration tool differed quite a bit from normal administrative and business application development, but he still found BaRE as a suitable and ready-to-use method to start with systematic RE, and expected it to speed up the adoption phase considerably. This expectation was well realized in Phase 2 when the requirements development work was started with only a few days of learning and adaptation work. Thus the senior software engineer considered ready-to-use methods beneficial for companies by providing an easy and quick way to start improving practices. Despite of this he had not seen any other ready-to-use method before. As a central benefit of a thought-through process he found the fact that it ensured things getting done in a right way and order.

The BaRE method was found to be ready-to-use and provide an infrastructure that was also ready-to-use. The only major obstacle before this enlightenment was learning what each section in the RD template meant and how each element in the method fitted to the whole picture. As an example of this the senior software engineer mentioned that he actually tried to develop his own RD template, but soon realized that the original one was better. According to his experiences, the method eased the development work considerably.

The **fit-for-need** of the BaRE method in Company B was good. By the end of Phase 2 nobody had noticed anything missing from it, even though three people studied the method in the adoption phase in detail and five others reviewed the developed requirements document. Also the number of extraneous parts in the method was limited and only a few techniques and parts in the RD template were omitted in the adaptation phase. For the requirements management practices, the BaRE method was used in a descriptive way and the resulting process was very similar to the original one, while in the requirements development the method was practically

Appendix 2. Case Study Descriptions

adopted as it was. However, it is important to bear in mind that once the RE practices mature in the company, adaptation work for needs-specific system administration tools is expected. As the last point, the possibility to adopt the BaRE elements in phases proved crucial in introducing it in Company B.

The **ease-of-use** of BaRE was found acceptable. A close examination of BaRE revealed its ease-of-use and raised the interest in adopting the whole method in use, as it appeared to provide a clear standard for doing RE in the company. As a whole BaRE was commented to look the better the more one understood it. The only problem was to get started with the method, but after that the method led the way forward. One important property to this end was the division of the document to multiple parts and putting the details in appendices. The provided documentation was found sufficient, except that a comprehensive example on a requirements document would have been needed. Otherwise the documentation was judged as understandable, clear, and logical. The overall systematic approach was found valuable, as it ensured that everything got done.

The **comprehensiveness** of the BaRE method proved sufficient for Company B. The adoption was started from the requirements management practices and tool before moving to the requirements development part, so that all the areas suggested in BaRE were finally covered. During the project only one essential requirement for system administration tools was discussed. This was reliability and how to measure it in practice.

Company B chose a freeware RM tool to support their process and used the BaRE method for the other **key constructs**. The BaRE RD template was first reviewed and due to a problem in completing it, an attempt to develop an own template was initiated. However, once the examples in the RD template were available, the BaRE template was taken in use with only minimal adaptation. The requirements development practices were also familiarized with, but after that the RD was developed without support from them. The requirements management process was defined and documented with common sense after the tool had been selected to support the process. The training in Company B was limited to self-study and discussions with the author of this thesis. In short the selected solutions in the company seemed to work well: the change management problems were solved and the RD template provided a good basis for software development. The BaRE method was considered as the company way to operate in the future and the only major difference to it was the established RM process. No standard training for the company was defined and from the BaRE practices only checklists were not used at all during the project.

Three **general lessons** were learned from implementing the changes. The first lesson was an issue the development manager realized as soon as the improvement efforts started – all the stakeholders in the company needed to be provided a general introduction to RE and process improvement topics. This was implemented by a presentation and discussion meeting with the author of this thesis, which seemed to clarify the situation. The next lesson was getting the new change management practices accepted in the company, since a previous effort in this direction had failed. The new features this time were a documented process including a feedback step, tool support for the process, and refusal to accept changes in any other way but through the system. All these features were announced as the standard way to work in the company from there on and the company management expressed clear support for the new practices. After some initial inertia the practices were accepted and everything worked fine towards the end of

Appendix 2. Case Study Descriptions

Phase 1. However, when the key people in the RE changed in the beginning of Phase 2, there was a setback to manual delivery of change requests to the senior software engineer. After a month this issue was raised in a company meeting and it was announced that the previously used electronic change management was the standard way to submit change requests, and it was taken in use again. Consequently in the end of Phase 2 it appeared as a real standard way of working in the company. The last lesson concerned the RD template and its adoption. The senior software engineer had ten questions on the RD template that were discussed with the author of this thesis. These questions could be categorized in two groups: abbreviations that needed to be expressed in full words and new concepts that had to be explained. After all the new concepts were explained with localized examples in the company domain, the senior software engineer started working on this first RD.

It was estimated that in case only the BaRE documentation had been available, the RE improvement effort would have succeeded to some extent, but not in the achieved scope, detail, and schedule. Thus the external support was found very valuable and necessary, especially due to the changes in the personnel. The author of this thesis acted first as a champion for the method, but for the senior software engineer the role changed to that of a change agent providing technical advice in a timely manner (Humphrey 1989, p. 31). The contacts with the author of this thesis were reported positive to the company also from the progress monitoring point of view, as they forced the personnel to try to do something also in the RE area.

As a whole BaRE was found central to the process improvement work in Company B. The senior software engineer stated that without BaRE the RE could still be in a miserable state in the company and if he would have had to devise the steps to improve the situation, the current practices would not be as good as they were now. He also promised to pass information about BaRE to people needing it since it worked so fabulously in their company.

1.4. Case C

Company C planned to test the BaRE method in full in a single project, but a failure to find a suitable one led to testing only the BaRE RD template. A Finnish translation of the template was done and it was used in two projects. With positive experiences from these projects, the template was revised and used in other two projects. These four projects and the RD template adoption process provided many valuable lessons from the workings of experienced software professionals in a small company focusing to bespoke system development and are described in this case account.

1.4.1. Background

Company C was a six-year-old software house living on bespoke systems development. Only one of the employees was not involved in software development, and even the managing director tried to do his share of the development; all the other people were fulltime software developers. The application domain for C was administrative and business applications which were often connected to mobile devices for some additional features (i.e., ABA/md in Table 23). Thus the software type was custom software although there was a desire to move to standardized components and even develop own software products.

Appendix 2. Case Study Descriptions

Company projects varied a lot, but as a rough estimate of their **typical project** the characteristics in Table 23 were estimated. Most of the projects had two or three developers in addition to a project manager who also took care of the RE. As typical time frames it was estimated that the RE would take one month and the development five months after that. For the effort, RE was estimated to take about one person month and the following development about 18 person months.

Table 23. Typical project characteristics

Project characteristics		Typical
Team size [people]	RE	1
	Development	2-3
Duration	RE	1 MO
	Development	5 MO
Effort	RE	1 PM
	Development	18 PM
Application Domain		ABA/md

In Phase 1 there were two **key participants**: a software architect (C1) and a project manager (C2). The software architect was among the senior people in the company and among the forerunners in process improvement activities. The project manager joined the project a little bit later and started looking at the BaRE method to be able to use it in his projects. The software architect had a Master's degree, while the project manager had his Master's degree still under development; in the beginning of Phase 1 he had completed the studies required for a Bachelor's degree. Both of them had 14 years of experience in software development. In Phase 2 the software architect focused on design and development tasks and did not do RE at all. However, another project manager (C3) did a project parallel to C2 and thus two projects were run simultaneously also in Phase 2. The project manager C3 had 12 years of experience and a Master's degree. These key people and their competence levels are summarized in Table 24.

Table 24. Key people in the BaRE evaluation project

Person	Title	Education	Experience [years]	Competence level				
				N	T	S	P	E
C1	Software architect	M	14	0	0	0	5	11
C2	Project manager	B*	14	0	1	4	6	5
C3	Project manager	M	12	0	1	2	7	6

Discussion on RE-related **problems** led to more general process problems where three issues were identified. These were lack of time to document processes, limited sharing of software development and process knowledge, and different working practices between different people. The processes themselves were considered to be clear, due to the long experience people had, but a lack of documentation inhibited their wider use and establishment of standard practices in

Appendix 2. Case Study Descriptions

the company. As a minor detail it was mentioned that qualifying as a training job location for university students provided an additional motivation for the documentation effort.

For Phase 1 of the BaRE evaluation project two **goals** were set: defining a process for requirements development, and creation of requirements document templates. Since achieving these goals in Phase 1 in full failed, the goal of Phase 2 was to get the template in larger scale use in the company (Table 25).

Table 25. Goals for the BaRE evaluation project

Phase	Goal
1	Define requirements document template(s). Define requirements development process.
2	Spread knowledge about the new RD template and get it in a wider scale use in the company.

There were no process guidelines defined for **software development**, as each project was different, and suitable practices were decided case by case with customers. However, there was a general feeling that things had been performed in a fairly organized way and standard solutions had been established whenever appropriate. E.g. changes had been managed with the StarTeam change management system with one customer, while some other customers had agreed on them in meetings, and even less formal procedures had been used occasionally. StarTeam was also a standard configuration management solution for one customer, while CVS software was used for version management in internal projects. StarTeam and CVS were the only special purpose software tools used in the company, and specialization in roles was similarly limited. Most of the people were called software developers, with the exception of one software architect and two project managers. A general feeling was that experience and interest in different areas was much more important than titles or explicit roles.

Considering the **software and requirements engineering practices**, it was clear that the company relied on the expertise of a few key people and, consequently, it was estimated to be on initial level in the CMM. Well in line with this the company got seven points in the Lightweight REAIMS Top Ten assessment. Only informal reviews had been conducted in a normal manner for the past six months, while five other practices had been followed on the discretion of the project manager.

The BaRE project was the first real **process improvement** activity in the company. There had been some initial efforts to improve the practices, but they soon got halted as the job was so difficult that tackling it appeared unfeasible with the available resources. In general the idea in process improvement was to follow the tSoft project and try to hang on with the activities in it, e.g. the BaRE evaluation project. As a whole the process improvement was not considered as a luxury but a necessity for the daily software development in the company.

1.4.2. Improvement Actions

In the beginning of the BaRE evaluation project, a project was sought where BaRE could be applied in full. The efforts to find such a project failed, however, and a full scale testing of

Appendix 2. Case Study Descriptions

BaRE was abandoned. Despite of that the BaRE RD template was interesting from the daily work point of view and thus it was taken in use. Since Company C had Finnish customers, a translation of the template was needed and done by the project manager C2. The software architect C1 got one version of this translation and used it in his own project. In Phase 2 the requirements documents developed in Phase 1 were evaluated, and a common template was developed by the project manager C3 together with C2 and C1. Thus all the people doing RE in the company were involved in the RD template development work. Despite of this, the template was not named as a company standard template yet. No other BaRE elements were utilized in the company; the project manager C3 studied the material, but he did not explicitly utilize it in his project due to strict time constraints.

The BaRE **RD template** had three novel features to Company C: introduction, business viewpoint, and providing a general overview to the system before going to actual requirements. The two separate documents developed in Phase 1 were not discussed between the projects. This outcome was not a planned one, but resulted from separate and busy projects; when the data for the BaRE evaluation project was collected in the end of Phase 1 this situation was revealed to all participants. In Phase 2 the project manager C3 familiarized himself with the BaRE method and a conscious effort to develop a common RD template was initiated. The projects run by project managers C2 and C3 kept each other updated about their RD developments, and in case of bigger issues the software architect was involved in the discussions, and decisions were made to keep the RD template common. Naming this template as a company standard was considered premature and a proper review process and common decision were set as prerequisites for such an appointment.

The **training** in Company C was in practice limited to self-study. The software architect C1 attended the tSoft seminars and the managing director participated in one seminar. The project managers C2 and C3 relied on the written material on BaRE; C3 studied it in full but the others used it only as a reference manual and did not really read it through. No company specific training or presentations on RE were provided and before the present project no training in areas interesting from this study point of view was provided. The specialist support was limited

Table 26. Training and information sources utilized by different people during the different phases

Information Sources	Phase 1			Phase 2			
	C1	C2	C-Others	C1	C2	C3	C-Others
BaRE Guide	x				x	x	
BaRE Templates	x	x			x	x	
Practical RE in Short							x
RE Presentations	x		x				
RE Courses							
Specialist Support/Consultation	x	x				x	
Software Engineering Training	x						
Application Domain Training							
Related Literature							

Appendix 2. Case Study Descriptions

to commentary on requirements documents and some discussions with the author of this thesis (Table 26).

No other parts of the BaRE method were utilized explicitly. Most of the BaRE requirements development process steps were taken when the original template was synchronized with the process, and completing the document requires addressing the steps in some way. In the same vein the other BaRE processes and techniques are very basic ones and get addressed if requirements development and management are done at all.

The improvement actions in Company C started slowly, but towards the end of the project a number of improvements had been achieved. In the RE area the improvements were limited to the development of an own RD template that was used in two projects, but plans to standardize it in the company use were in progress. In the end of the present project it was also learned that similar common templates were under development for contracts and recording changes. Another activity was the development of initial process guidelines that were put to the intranet, so the process improvement work appeared really to have started in Company C during the project. As the key benefits of participating in the BaRE evaluation project it was noted that the scope for the improvement effort had enhanced considerably and the adoption was both easier and faster due to the help received. The improvement actions were done as part of normal projects, which will be looked at next.

1.4.3. Requirements Development Projects

Company C ran four projects using an RD template based on the BaRE template during the BaRE evaluation project. Since the documents were considered project specific and no company specific template was established yet, the following discussion focuses on the documents, not on any template. Before looking at the documents the projects are described.

Both Phases 1 and 2 followed two different projects. The first project C-P1 was a small one intended to develop an application to integrate two systems with a standard communication protocol. The project was run by the software architect C1, who was also the only worker in this six-person-week project. The project was completed in six weeks, except for the integration phase which took six months to complete due to various business decisions. Project C-P2 developed an administrative application to help in project management. In addition to internal needs, it was estimated that the application would have a wider-spread interest and thus it could serve as a basis for customer-specific adaptations in the future. In project C-P2 the first version of the application was developed and released. The project was run by the project manager C2 and two other people did the actual application; altogether the project lasted three and a half months, during which the requirements were developed mostly parallel to development. The total effort for the project was about six person months. These two projects were done in Phase 1, and the project manager C2 moved on to project C-P3 in Phase 2. In this project an application to manage product data and to use it in business was developed. Project C-P4 was run by the project manager C3 and it was fairly similar to project C-P3. The biggest difference between these applications was the application domain; the similarities are seen well in the project summary in Table 27. Both projects moved from design to implementation phase in the end of Phase 2.

Table 27. Key characteristics of the projects

Project characteristics		Phase 1		Phase 2	
		C-P1	C-P2	C-P3	C-P4
Team size [people]	RE	1	1	1	1
	Development	1	2	2-3 (e)	2-3 (e)
	Integration	1	-	-	-
Duration	RE	1 W	4 W	6 W	5 W
	Development	5 W	3.5 MO	4 MO (e)	4 MO (e)
	Integration	6 MO	-	-	-
Effort	RE	1 PW	5 PW	3 PW	5 PW
	Development	5 PW	5 PM	4.5 PM (e)	4.5 PM (e)
	Integration	3 PW	-	-	-
Application Domain		ABA/md	ABA	ABA	ABA
Project Status		Released	Released	Designed	Designed

Project C-P1 was so small that it basically followed the waterfall process model: the first RD version (Baseline 1 or B1) was first completed and software was developed based on it. The document was updated after the implementation phase to include the few changes that took place during the development (version Baseline 2, B2). The template for the document was an early translation of the BaRE RD template, and thus the document headings changed between the different document versions. The document contents did not change much during the development, but a number of new requirements were identified. The key document properties are shown in Table 28, which summarizes all the requirements documents developed in Company C during the present project. The customer in this project was not very technically oriented and the only comment they had on the RD was that it could have included more of a customer viewpoint. The software architect had already realized this from looking at the finished document.

Project C-P2 followed a more parallel approach, and the developers started working with a RD which was the result of only about three working days (B1). Consequently the document changed a lot, and e.g. term definitions had to be added to the document in three days, since misconceptions disturbed the development too much. The final document version was updated after the software had already been released; in this project the implemented software was also released without separate testing or integration phase, so Baselines 2 and 3 were the same (Baseline 3 or B3). Also in this project the customer did not really provide feedback to the requirements document but appeared happy with the document they received.

Projects C-P3 and C-P4 utilized a common RD template revised from the documents developed in Phase 1. The comparison of these two documents revealed only few minor differences that could be considered either as project specific adaptations of the template or typos. The inclusion of typos here is due to the fact that the projects C-P1 through C-P3 used MS-Word as their documentation tool, while C-P4 used LaTeX, and thus there were two separate document templates after all. Both projects had three appendices: C-P3 had 23 pages

Appendix 2. Case Study Descriptions

of appendices and C-P4 34 pages. By the beginning of the implementation phase, project C-P4 experienced only three changes and C-P3 had 11 refinements in the Baseline 1 documents.

The C-P3 project differed from all the other projects since it started with a previous RD on the planned application, and the RD template contents were accepted before the actual work started. Namely, the customer had contracted the RD development earlier to another company, but as they found the resulting RD incomprehensible, Company C was contracted to do it again. The project manager C2 stated that for him this previous RD created actually more confusion than help; the customer, on the other hand, commented on the new RD provided by Company C that they actually understood it with a single reading. In project C-P4 no feedback was provided by the customer on the RD, but the project manager C3 interpreted this lack of criticism as a sign of satisfaction.

Table 28. Properties of the produced requirements documents

Requirements Documents		Phase 1				Phase 2	
		C-P1		C-P2		C-P3	C-P4
Available versions		B1	B2	B1	B3	B1	B1
Page count contents/total		10/14	12/16	16/20	29/33	23/28	14/19
Appendix count		0	0	0	0	3	3
General	Identical	0	0	0	0	1	3
Properties	Similar	0	0	1	1	3	1
	Partly similar	1	1	0	0	1	2
	Different	5	5	5	5	2	1
	Added	0	0	0	0	0	0
	Omitted	3	3	3	3	2	2
	Count	6	6	6	6	9	9
Document	Identical	19	20	12	12	15	14
Headings	Similar	15	18	24	25	22	21
	Different	13	10	7	6	2	2
	Added	0	1	0	2	0	0
	Use case name	0	0	6	22	4	0
	Omitted	4	3	8	8	12	14
	Count	47	49	49	67	43	37
Contents for	Group header, empty	6	7	9	9	8	9
Headings	Not Available	9	0	19	12	0	0
	Available (unchanged)	32	24	21	4	35	28
	Added	0	2	0	25	0	0
	Extended	0	15	0	14	0	0
	Modified	0	1	0	3	0	0
	Removed	0	0	0	0	0	0
	Count	47	49	49	67	43	28

Appendix 2. Case Study Descriptions

The first document versions in projects C-P1 and C-P2 were discussed with the author of this thesis, which clarified some misconceptions about the original template and led to changes in the later versions. In Table 28 a *Group header, empty* -count is provided only to be able to match the numbers in the table; otherwise it is not considered meaningful here. The decreasing number of headings without contents (i.e., *Not Available* in Table 28) and the increasing number in the *Added* line show that new requirements were discovered during the project. The *Modified* and *Extended* lines indicate changes in requirements or in their representation. The *Available (unchanged)* line indicates in the first document version the number of headings having contents, and in the later version the number of unchanged headings; the decreasing number in this line indicates also changes after releasing the first version of the document.

In the common template used in projects C-P3 and C-P4, the differences to the BaRE template got even smaller. Concerning the general properties the cover page and document template looked a bit different, no tables were shown in the documents and context diagrams were the only diagrams in the documents. Even though table *frames* were removed from the documents, the requirements followed a standard format including an identifier, actual requirement, and in many cases a description of the requirement; these fields were also tagged systematically with headings. The two later projects also started using appendices; the three appendices both documents had, each had a glossary and use case descriptions. The additional appendices were report examples in C-P3 and an interface specification in C-P4. The use case names were dropped from the table of contents even though in C-P3 four topic areas for use cases were included as sub headings. An own addition to use case names was the inclusion of create, read, update, and delete (CRUD) operations after the actual names. As a whole the number of different heading names dropped to two, so the templates were very similar except for the fact that about one fourth of the suggested headings were dropped.

1.4.4. Changes

The biggest change in Company C during the present project was that co-operation in the company got much closer. Before the project everyone was struggling alone with the template he used, communication was much rarer and it often took place only after the document was completed. Such late feedback resulted often in minimal changes, since the document was already considered ready. Now discussion was much easier, as there was a common context – the RD template with an increasingly common terminology – and the discussions took place during the development phase. This improved and increased the common efforts to improve the shared template, and made it possible to discuss also project specific documents on the fly. Otherwise the changes in the company were marginal: one new trainee was hired, bespoke development was still the major project type even though the component development had started well, no new methods or tools were acquired, except for the internally developed tool to support project management (C-P2), and no specialists had been named.

The **Lightweight REAIMS Top Ten** assessment showed over 50% increase in the point gain but the practices were still in general on initial level. The changes took place only in four practices and each practice rose one level higher: a template was introduced for both requirements descriptions and RD, each requirement had a unique identifier, and developed documents appeared to be written with ease of change in mind (Table 29). All these changes took place with the common RD template developed in Phase 2. The SPICE RE process capability assessment result in Company C was the following: “A software requirements

Appendix 2. Case Study Descriptions

management process assessment was performed and level 1 was achieved. The assessment covered 8 base practices and most practices were fully or largely achieved. The most significant omission on level 1 base practices was lack of measurable quality requirements and lack of procedures to handle requirements change management and traceability; thus these two practices were achieved only partly. Level 2 assessment covered 8 management practices both in performance and work product management attributes. The performance attributes were achieved in general largely, while the management practices appeared to be less well addressed and the management attributes were achieved only partly” (Saastamoinen and Tukiainen 2003c, p. 18).

Table 29. Lightweight REAIMS Top Ten assessment point gains in different phases. The symbols mean the following: ‘-’ Never applied, ‘O’ Applied at the discretion of the project manager, ‘D’ Normal use, and ‘●’ Standard use in company

Lightweight REAIMS Top Ten Guidelines	Pre-BaRE	Phase 1	Phase 2
Define practices for RM	O	O	O
Define a standard document structure	O	O	D
Define unique identifier for each requirement	O	O	D
Define standard templates for requirements description	O	O	D
Define validation checklists	-	-	-
Organize informal requirements reviews	D	D	D
Use language simply, consistently and concisely	O	O	O
Make the document easy to change	-	-	O
Use checklists for requirements analysis	-	-	-
Plan for conflicts and conflict resolution	-	-	-
Total Point Gain	7	7	11

In the company **infrastructure** only the RD and requirements templates were established in normal use as mentioned above. Thus the infrastructure as a whole could not be considered very comprehensive, as shown in Table 30.

It is evident from the assessments above that also the **use of the practices** in Company C did not change much (Table 31). An RD template was used in all the four studied projects, and a process was used in change management and reviews in some of the projects. Finally, some of the techniques suggested in BaRE were utilized in all the company projects. The project manager C3 noted that even though no explicit requirements development process was used, the RD template did provide guidance in the work. The project manager C2, on the other hand, noted that the changes were not managed in a systematic way and the practices changed when the person to collect the change requests changed. The changes specific to RD are discussed in detail in Section 1.4.3 (Requirements Development Projects).

The explicit **goals** were not achieved in full even though they were few in number. However, a common RD template was developed and it was used increasingly in the company. The project

Appendix 2. Case Study Descriptions

Table 30. Infrastructure in the different phases. The options were the following: Not available, Discrepant artifacts exist, Common artifact available in company, and Standard artifact available in company

Element	Pre-BaRE	Phase 1	Phase 2
Templates			
Requirements description	Discrepant	Discrepant	Common
Requirements document	Discrepant	Discrepant	Common
Change requests	Not available	Not available	Not available
Processes			
Requirements development	Not available	Not available	Not available
Change management	Discrepant	Discrepant	Discrepant
Review	Discrepant	Discrepant	Discrepant
Techniques			
Requirements development	Discrepant	Discrepant	Discrepant
Change management	Discrepant	Discrepant	Discrepant
Checklists			
Requirements engineering	Not available	Not available	Not available
Change management	Not available	Not available	Not available
Training material			
Requirements engineering	Not available	Common	Common
Software engineering	Not available	Not available	Not available
Application domain	Not available	Not available	Not available
Methods			
Requirements engineering	Not available	Not available	Not available
Software engineering	Not available	Not available	Not available
Tools			
Requirements management	Not available	Not available	Not available
Change management	Not available	Not available	Not available

manager C2 considered the new template as a much better one than their previous ones and he was confident that only one more iteration with the template would result in establishing it as the company standard template. Table 32 summarizes the main achievements of this project.

The **problem areas** did not change during the present project. In the beginning the software architect stated that distributing the knowledge to other people was a key issue in the company, whereas in the end of the project the project manager C3 found common processes and standardization of the practices important. Since the RE practices had in fact become much more unified during the project, and two similar requirements documents had been developed, this problem area was addressed in this project with positive results.

The **next steps** were not yet clearly defined. Development of similar common contexts for other software engineering areas was found important, and the project manager C2 named change management as a candidate for next efforts. A desire to develop an own standard

Table 31. Use of the practices in the different phases

Element	Pre-BaRE	Phase 1		Phase 2	
		C-P1	C-P2	C-P3	C-P4
Templates					
Requirements description	Partly	Used	Used	Used	Used
Requirements document	Partly	Used	Used	Used	Used
Change requests	Not	Not	Not	Not	Not
Processes					
Requirements development	Not	Not	Not	Not	Not
Change management	Not	Not	Used	Not	Used
Review	Not	Not	Used	Used	Used
Techniques					
Requirements development	Partly	Partly	Partly	Partly	Partly
Change management	Partly	Partly	Partly	Partly	Partly
Checklists					
Requirements engineering	Not	Not	Not	Not	Not
Change management	Not	Not	Not	Not	Not
Training					
Requirements engineering	Not	Partly	Partly	Partly	Used
Software engineering	Not	Not	Not	Not	Not
Application domain	Not	Not	Not	Not	Not
Methods					
Requirements engineering	Not	Not	Not	Not	Not
Software engineering	Not	Not	Not	Not	Not
Tools					
Requirements management	Not	Not	Not	Not	Not
Change management	Not	Not	Not	Not	Not

requirements development process existed also, and it appeared only a question of time when it would be done. However, new projects were planned to start in a few months and, whenever RE was a part of them, the use of the new common RD template was considered worthwhile.

Table 32. Main achievements

Phase	Achievements
1	A Finnish RD template defined based on the BaRE RD template and tested independently in two separate projects.
2	An RD template defined as a co-operative effort in the company. Defined RD template used in two different projects.

1.4.5. Benefits

The biggest benefit from the BaRE evaluation project was found to be that actual development work had been started and the results had been adopted in normal use in the company. The key feature of the BaRE method that helped to achieve these benefits was the fact that it was really lightweight and did not require any massive training program or introduction to get started with. This, again, reduced the overall effort required in the process improvement work, made it possible for multiple projects to utilize the practices simultaneously, and thus, little by little, it was expected that the new practices could be established as a company standard automatically.

A common context for RE was considered to provide three important benefits for the company: the common terminology it provided ensured that terms actually referred to the same thing irrespective of projects or people, it made comparing things easier, and it eased discussions since the reference points could be identified quickly and reliably. In addition to these internal improvements the common terminology also eased the external contacts with customers and a partner company in similar ways. So far the changes had already allowed definition of a general project concept, the meeting practices had become more similar, and discussions on RE could be conducted even during coffee breaks.

1.4.6. Costs

Company C's investments in the process improvement were modest. Since neither tools nor training was purchased, the only cost was the time used in process improvement related activities (Table 33). In this area the biggest cost was the RD template development work, and the second biggest cost was participation in the tSoft seminars, training, and the SPICE evaluation. The software architect C1 participated in this project practically only in Phase 1 and the project manager C3 only in Phase 2. In total the company invested less than four person weeks to improvement efforts during the BaRE evaluation project.

Table 33. Use of time in hours for RE and process improvement-related activities

Person	tSoft	RE Info	Meetings	Training	Study	RE Dev	Total
C1	8	6	7	15	3	6	45
C2	7	0	7	0	3	23	40
C3	0	0	3	0	2	40	45
C-Others	11	1	2	0	0	0	14
Totals [h]	26	7	19	15	8	69	144

1.4.7. Lessons Learned

Company C utilized only the BaRE RD template in their process improvement efforts, but managed to provide many interesting lessons thereby. In the following the standard questions are discussed on the basis of the experiences with the RD templates, and supplemented with some more general lessons from the project. Before going to these topics it is, however, worthwhile to note that a standard process and document template for RE were considered important for project planning and tracking already in the outset of the present project. That is,

Appendix 2. Case Study Descriptions

Company C wanted to improve their project management even though it was not considered as a problem in the company. The lessons the key participants learnt from adopting and adapting the RD template were practical ones: only one RD template was selected for further development, templates for requirements descriptions were used, standard operating modes were added to the template, use cases were moved to appendices, CRUD operations were documented for every use case, etc. Consequently, looking at the developed RDs in the end of the project, it seemed a fairly simple a task to move much of the RD contents to a requirements database and start reusing the requirements from there instead of the current document level reuse.

The **ready-to-use concept** was found valuable in Company C. A key property of the concept was considered the possibility to assess a method quickly by looking how it solves problems an expert knows from experience. Another important benefit from the concept was the lightweight adoption process in provided. Namely, Company C had started analyzing their working practices before the BaRE evaluation project, but all the previous improvement efforts had been buried under the daily workload due to the large efforts they required. When a quick assessment of BaRE seemed to provide solutions to their problems and the method itself seemed easy to adopt, a decision to test the method on an as-is basis in practice was made. Lightweight methods and adoption processes were considered important for small organizations, as heavyweight adoption and adaptation results in rejection of improvement efforts from the outset and in sticking to existing practices. This follows from the fact that small projects may even be implemented by the time a heavyweight development process takes to start. It was also stated that ready-to-use methods for e.g. software engineering, testing, and change management would be interesting from the company's point of view.

The BaRE method was not taken in use in full during the present project. One of the reasons to consider it in the first place was that it appeared like a suitable project independent tool that would provide immediate help in many areas. Another reason to consider BaRE as a complete method without changes followed from the fact that it was a thought-through model to do RE, and modifying it was not a goal in itself. The only other method that appeared to be ready-to-use was RUP, but the confidence in this conception was not very high, and consequently no actions had been taken to adopt RUP. In the lack of suitable projects, also the adoption of BaRE was limited to the RD template, which appeared to provide the best cost-benefit ratio to the daily work.

The BaRE RD template was found **fit-for-need** in Company C's projects. Both the application domain and the company needs seemed to fit well for the approach taken in the method, and as no clear needs for modifications were noticed, it made sense to use it to develop customer documents. The received customer feedback on the RDs was minimal, except for the project C-P3 where the customer considered their RD understandable. The software architect found the template reasonable, which was not the case with some of the templates he had seen in his career. Consequently, he stated that without a doubt, the BaRE template was valuable especially to small and medium software houses that do not have resources for thorough RD development efforts. Well in line with these opinions he found the inclusion of business and product position statements a good solution, as separate vision documents were not written in Company C's projects. In the closing interview for the present project, after getting first hand experiences from their templates, neither one of the project managers could point to any real problems with the latest one, but considered it suitable for their application domain and needs.

Appendix 2. Case Study Descriptions

The **ease-of-use** of the BaRE method received dual comments. Both project managers stated that the method appeared simple, but following it forced to get back to the instructions and to consider where different topics actually go in the document. So the method had something that made it appear simple, but in practice there was something that created uncertainty at least during its first use. In some instances the English language could be pointed out as the reason for problems, but not always. Another specific example mentioned was placing all the new terms in a glossary appendix instead of the actual RD. The problem with this approach was that earlier terms had been defined as one of the first things in a project and including them, respectively, in the beginning of an RD made a lot of sense. Putting terms in an appendix required a second thought before the new structure was just accepted. So both project managers stated that the BaRE method was not trivial, but required real investment in learning. The software architect mentioned already in Phase 1 the initial problems other people had with the adoption, but he himself never appeared to have similar problems.

It was believed that having a full unified example of an RD would have eased the method adoption quite a bit. Otherwise the documentation was considered sufficient, even though some more descriptive text would have made the documentation and especially the RD easier to understand. As it was, the project manager C3 stated, the understandability of the method suffered from the puritan approach it had, while adding some more explanations and words would make the documentation self-sufficient. In general the documentation was found logical and comprehensive. The only problem in the adoption phase was figuring out where to put some specific requirements; after these problems were solved no further problems were encountered. Neither one of the project managers participated in any tSoft or RE presentation and the project manager C3 believed that such a presentation would surely have made the adoption task easier. One specific problem with the RD template was mentioned by the software architect who said that documenting the requirements by an overall description supported with specific requirements was originally hard, as he was accustomed to describing software through requirements. However, after getting acquainted with the new way of writing he liked it.

In general the **comprehensiveness** of the BaRE RD template was found good with one extension request. Namely, it was hoped that already the RD template would include an introduction explaining what the document contains and why. No other omissions in the template was identified; one customer wanted to document their own notices about the software and its adoption, but these fitted well in the general requirements topics the template included. In the same vein no useless parts in the RD template were identified. Even though it was not always clear from the beginning what to write under each heading, they provided a good indication what should be addressed in the document. The software architect admitted that the template included topics that really should be addressed in an RD, but no previous template he had seen had included them, especially the goals and stakeholders. To the question of whether some performance requirements could be omitted he said that none of them could be left out. All the performance aspects had to be mentioned in the document in order to show the customer that these topics are covered by the analysis and the document. In the end of his project he actually indicated example requirements from his project for each performance requirement category in the BaRE RD template.

Testing of the **key constructs** in BaRE was practically limited to the RD template. The BaRE Guide was read through or skimmed by multiple persons, but little of it was taken in use in the

Appendix 2. Case Study Descriptions

company. Some of the suggested requirements development and management practices were followed, but this came from the fact that they were basic techniques used in a proper RE anyway. No tool to support the practices was acquired and training was limited to self-study and some discussions with the author of this thesis. Also little infrastructure outside the RD template was used, and even the used ones were not declared as a company standard. Considering the working practice, on the other hand, little change could be seen.

The work with the RD template resulted in a number of more **general lessons**. Thus the rest of this section describes the RD template adoption and adaptation processes, the role of a written requirements document, the benefits of having one, and the role of the author of this thesis in this study before the closure of this section.

The **adoption process** in Company C was a very practical one. Two people developed their own requirements documents based on a similar template independently in different projects, and after realizing this situation they shared their experiences and found the template to work well. Consequently, a more general discussion took place in the company and yet another person was convinced to test the template in his project. Two revisions of the RD template were done, and both revisions were tested in two projects running practically simultaneously. After the second revision, discussions about a company standard RD template were started, but a third revision and a common decision about the standard were considered essential to really establish a common practice in the company. This final revision was expected to be developed as a part of the next company projects.

The translated version of the RD template provided insights into the RD template **adaptation process**. Namely, the translation introduced some unintentional changes in the template and one incident is reported in more detail here – the Operation Types. In one project the original heading was translated to “Operations” and use case descriptions were expected to be written under it, while the other project used the translation “Way of Using” describing internal and external users with differences in their user interfaces. Even though the project manager C2 did the actual translation work, the software architect took an intermediate version of the template and modified it as he saw fit himself. Some of the terms in the template were ambiguous to both, but no serious attempts were made to understand them. Due to the time constraints, quick decisions had to be made, and in the lack of a ready-to-use RD template, a document with sketchy headings was put forward. In the discussions with the author of this thesis it was admitted that problems with the RD template were focused on headings referring to concepts that were not properly understood. These concepts and headings were, however, readily understood with examples from the BaRE RD template explained by the author of this thesis.

The **role of a written requirements document** was found crucial for software development projects. The software architect mentioned that a written RD is important in a project even though a customer is not interested in it, while with a well-versed customer it is a necessity. In his own six-person-week project the RD was also important since the effort was significant with respect to the company size. The project manager C2 supported these views and stated that he could not figure out a project where a requirements document would not be needed.

Both the software architect and the project manager C2 could name **benefits of having a requirements document** in their projects. The software architect told that for him the RD had served as a checklist where he first wrote all the things he had to do and then he actually

Appendix 2. Case Study Descriptions

checked the items in the document he had completed with a pen. This approach helped him to bind the solution and to speed up the development. The project manager backed up these views by telling that an RD is one of the most important documents in a software project – a bible – that can always be referred to. He reported that he had used the RD in almost every meeting to check things up before commenting on detailed questions about implementation. Another of his practical experiences was the inclusion of term definitions in the RD. The definitions were not included in the document that was first given to the developers, but they had to be added there in only three days when they appeared necessary for the communication between him and the two developers.

All the key participants in Company C considered **the role of the author of this thesis** important for the improvement effort. The author of this thesis was found very important in solving some technical issues quickly in the early phases of the adoption, even though the support was in general quite limited. As a whole the author of this thesis had the role of both a champion and a change agent in Company C (Humphrey 1989, p. 31). The fact that someone outside the company was interested in the improvement actions was also considered to have had a positive impact on the effort.

As a whole the BaRE method was considered helpful in introducing and improving RE practices in Company C. The future plans in RE included continued RD template improvement and studying possibilities to extend the adoption of the BaRE method to include also other parts of it as standard practices in the company. Both project managers stated that their experiences from the BaRE method were positive and promised to pass a word about it to anyone needing help in improving RE practices.

REFERENCES

- Curtis, B., W. E. Hefley, and S. A. Miller (2001). People Capability Maturity Model (P-CMM). Maturity Model CMU/SEI-2001-MM-01. Pittsburgh, PA, USA, Software Engineering Institute: 735 p. Available at <http://www.sei.cmu.edu/publications/documents/01.reports/01mm001.html>.
- Curtis, B., W. E. Hefley, S. A. Miller, and M. Konrad (1997). "Developing Organizational Competence." Computer **30**(3): 122-124.
- Humphrey, W. S. (1989). Managing the Software Process. Reading, Massachusetts, Addison-Wesley.
- ISO/IEC TR 15504-1 (1998). Information Technology - Software Process Assessment - Part 1: Concepts and Introductory Guide. ISO/IEC Ed., International Organization for Standardization and International Electrotechnical Commission.
- Saastamoinen, I. and M. Tukiainen (2003a). Company A: Assessment Report (Confidential). Joensuu, University of Joensuu: 73 p.
- Saastamoinen, I. and M. Tukiainen (2003b). Company B: Assessment Report (Confidential). Joensuu, University of Joensuu: 70 p.
- Saastamoinen, I. and M. Tukiainen (2003c). Company C: Assessment Report (Confidential). Joensuu, University of Joensuu: 69 p.
- sfrm (2002). Software for Requirements Management. http://cs.joensuu.fi/pages/saja/se/sfrm_mainos.html. Accessed June 17, 2002.

Appendix 2. Case Study Descriptions

Software Engineering Institute (1995). The Capability Maturity Model: Guidelines for Improving the Software Process. Reading, MA, Addison-Wesley.

Sommerville, I. and P. Sawyer (1997). Requirements Engineering: A Good Practice Guide. Chichester, England, John Wiley & Sons.

Vickers, A., A. Mavin, and H. May (2002). "Requirements Engineering: How Do You Know How Good You Are?" IEEE Joint International Requirements Engineering Conference, Industrial Presentations, University of Essen, Germany, IEEE Computer Society. pp. 91-99.

Competence Evaluation Form

Name:

Date:

Company:

Title:

Education:

Working Experience [yrs]:

Competence in Using BaRE

The skills needed to use BaRE effectively can be acquired, e.g., in the following ways:

1. Practical level courses in universities (e.g. use cases, reviews, etc.)
2. General knowledge (e.g. introspection, negotiation meetings, reading, etc.)
3. Self study (basically all skills with appropriate literature)
4. Special training courses (e.g. RM tools use, use cases, etc.)

Requirements Engineer's Competence

Level of competence	Characterization
Novice	New to topic in question
Trainee	Knows theory
Supervised practitioner	Can do; supervision and pre-reviews are advantageous
Practitioner	Has done; works independently and participates in peer reviews
Expert	Proven ability; experienced in different domains and adaptation

The level of competence in each area is defined based on the lowest competence in each area considering the detailed topics for each area.

The competence areas reflect directly the BaRE version v1.0: the techniques are the ones described in the BaRE Guide and the specification topics are taken directly from the BaRE requirements document template. Thus modifying the method should be reflected in the detailed topics below, as well.

General Competences

N T S P E

- Software Engineering (working experience, projects done, products shipped)
- Requirements Engineering (working experience, requirements development projects done, requirements documents written)
- Application Domain (working experience)

Competence in Processes

N T S P E

- Software Development
- Requirements Development

Appendix 3. Competence Evaluation Form

- Change Management
- Evaluation/Review/Inspection

Competence in Techniques

Elicitation

- Apprenticing
- Document Studies
- Electronic requirements
- Interviews
- Introspection
- Meetings: Brainstorming
- Meetings: Negotiation
- Meetings: Workshops
- Reuse Requirements
- Use Cases

N T S P E

Analysis

- CRUD Matrix Development
- Define System Boundaries
- Define the System's Operating Environment
- Diagram Development
- Reviews

N T S P E

Validation

- Checklists
- Prototyping
- Reading
- Write User Manual

N T S P E

Requirements management

- Baselining
- Change Management
- Version Identification
- Maintain Change History
- Tool Support for RM

N T S P E

Documenting

- Two-Level Approach to Requirements Documentation

N T S P E

Competence in Specification

Business view

- Customer Problem
- Business Goals
- Stakeholders

N T S P E

Product description

- Product Purpose
- Product Position Statement
- Product Context
- System Interfaces
- User Interfaces
- Logical User Interface Characteristics
- Operation Types
- Product Functions as Use Cases
- Users

N T S P E

Application domain

- Important Domain Properties
- Business Process Description

N T S P E

Prioritization

- Apportioning of Requirements
- Scope of Initial Release
- Manual Operations
- Restrictions, Limitations, and Exclusions

N T S P E

Performance

- Performance Requirements
- Response and Processing Times
- Precision
- Capacities

N T S P E

Data

- Database
- Data Model

N T S P E

Software system attributes

- Reliability
- Availability
- Security
- Maintainability

N T S P E

Appendix 3. Competence Evaluation Form

- Supportability
- Portability
- Usability
- Scalability

Non-functional requirements

- Documentation Requirements
- Miscellaneous Requirements
- Likely Changes
- Copyright, Legal, and Other Notices

N T S P E

Miscellaneous

- Constraints
- Assumptions
- Dependencies
- Design Constraints

N T S P E

Requirements Document for *PReMa*

<The text conventions in this document are the following:

- *Example text is shown in italics.*
- *Instructions, like this section, are shown in italics enclosed with '<' and '>' characters.*
- *When possible, examples and instructions are also marked as hidden text.>*

Product Name	<i>PReMa</i>
Product Version	<i>1.0</i>
Document Created	<i>2002-03-25</i>
Document Modified	<i>2002-10-02</i>
Document Issued	<i>2002-10-02</i>
Document Version	<i>1.0</i>
Document Status	<i>Accepted</i>
Project Name	<i>RETICE</i>
Confidentiality	<i>Confidential</i>
Customer	<i>tSoft</i>
Preparers	<i>un</i>

Copyright © 2002 by Uolevi Nikula. Permission is granted to use, modify, and distribute this document internally. It may not be sold or used for commercial gain or other purposes without prior written permission.

This BaRE method document template version 1.0 has been developed as a part of the licentiate thesis of Uolevi Nikula, Lappeenranta University of Technology, Finland, in conjunction with the tSoft technology transfer program of the University of Joensuu, Finland. The template is currently under test use in tSoft projects and will be revised based on the experiences from industrial projects run under the tSoft program.

CHANGE HISTORY

Date	Version	Description	Author
2002-03-25	0.1	Created document	un
2002-04-12	0.2	Removed Viewpoints –topic as irrelevant	un
2002-04-12	0.2	Changed all ID field names to ID-only	un
2002-04-12	0.2	Added Table Text –text style	un
2002-04-12	0.2	Added examples as hidden text	un
2002-04-12	0.2	Added missing tables	un
2002-04-12	0.2	Changed SC (Snow Card) field name to DR (Detailed Requirement)	un
2002-04-12	0.2	Removed Change-appendices to their own documents	un
2002-04-12	0.2	Removed Data Dictionary from appendices for now	un
2002-04-12	0.2	Added App 5: Rejected Requirements	un
2002-05-28	0.2	Footer translated to English	un
2002-05-30	1.0D1	Changed versioning scheme	un
2002-05-30	1.0D1	Removed Status –column from Appendices table, no need anymore	un
2002-06-04	1.0D1	Reordered contents to reflect the process	un
2002-06-04	1.0D1	Added table for UI logical characteristics	un
2002-09-24	1.0D2	Added examples, minor restructurings	un
2002-10-02	1.0D3	Started using a cover page	un
2002-10-02	1.0A	Accepted the document	un

TABLE OF CONTENTS

Change History	1
Table of Contents	2
1 Introduction	4
1.1 Customer Problem	4
1.2 Business Goals	4
1.3 Stakeholders	4
1.4 Product Purpose	5
1.5 Product Position Statement	5
1.6 Document Overview	5
1.7 References	5
2 Overall Description	6
2.1 Product Context	6
2.2 Product Perspective	6
2.2.1 System Interfaces	6
2.2.2 User Interfaces	6
2.2.3 Logical User Interface Characteristics	7
2.2.4 Operation Types	7
2.3 Product Functions as Use Cases	8
2.4 Users	8
2.5 Problem Domain Description	8
2.5.1 Important Domain Properties	8
2.5.2 Business Process Description	9
2.6 System Requirements Allocations	9
2.6.1 Apportioning of Requirements	9
2.6.2 Scope of Initial Release	10
2.6.3 Manual Operations	10
2.6.4 Restrictions, Limitations, and Exclusions	10
2.7 Constraints	10
2.8 Assumptions	11
2.9 Dependencies	11
3 Specific Requirements	11
3.1 Performance Requirements	11
3.1.1 Response and Processing Times	11
3.1.2 Precision	12
3.1.3 Capacities	12
3.2 Data Requirements	12
3.2.1 Database	12
3.2.2 Data Model	13
3.3 Software System Attributes	13

3.3.1	Reliability	13
3.3.2	Availability	14
3.3.3	Security	14
3.3.4	Maintainability	14
3.3.5	Supportability	15
3.3.6	Portability	15
3.3.7	Usability	15
3.3.8	Scaleability	15
3.4	Other Nonfunctional Requirements	16
3.4.1	Documentation Requirements	16
3.4.2	Miscellaneous Requirements	16
3.4.3	Likely Changes	16
3.4.4	Copyright, Legal, and Other Notices	17
3.5	Design Constraints	17

APPENDICES

Appendix	Date	Version
1: Glossary		
2: Typical Computer Configuration		
3: Use Case Descriptions		
4: Detailed Requirements		
5: Rejected Requirements		

1 INTRODUCTION

1.1 Customer Problem

<Rationale for the new product, short history and events leading to the development of the new product.>

1.2 Business Goals

<Business goals that justify the purchase/costs of the software, expected benefits, savings, etc.>

ID	Description	DR
G1		Y/N

<i>Replace outdated platform</i>
<i>Integrate order documents and database</i>
<i>Use experience data for quotation</i>
<i>Support systematic marketing</i>
<i>Faster capture of cost data</i>
<i>Speed up invoicing</i>
<i>Reduce IT costs</i>
<i>Remove error sources</i>
<i>Observe deadlines</i>
<i>Reduce over- and undertime payment</i>
<i>Improve roster quality</i>
<i>We cannot do (survive) without it</i>

1.3 Stakeholders

ID	Stakeholder Type	Description	Responsibilities
ST1			

<Stakeholder examples >

<i>End-users of the system</i>
<i>Customers</i>
<i>Developers (design, coding, testing, documenting, quality assurance)</i>
<i>Maintenance staff (backups, restoring, bug fixes, enhancements, changes)</i>

<i>External bodies: regulators, certification authorities</i>
<i>Management, project sponsor</i>
<i>Business subject matter experts</i>
<i>Technology people</i>
<i>Marketing people</i>
<i>Product managers</i>
<i>Lawyers</i>
<i>Usability experts</i>
<i>Professional bodies</i>

1.4 Product Purpose

<Problem to be solved with the software>

1.5 Product Position Statement

<Product position statement fits to products but is not as suitable for bespoke system development projects>

For	<i><target customers></i>
Who are dissatisfied with	<i><the current market alternative></i>
Our product is a	<i><new product category></i>
That provides	<i><key problem-solving capability, that is, compelling reason to buy></i>
Unlike	<i><primary competitive product alternative></i>
We have assembled	<i><key whole product features for your specific application></i>

1.6 Document Overview

<Who/roles should read which parts of the document, where to find different kinds of information>

Reader Type	Sections to Read
<i>Manager/Designer/Tester/Customer/...</i>	

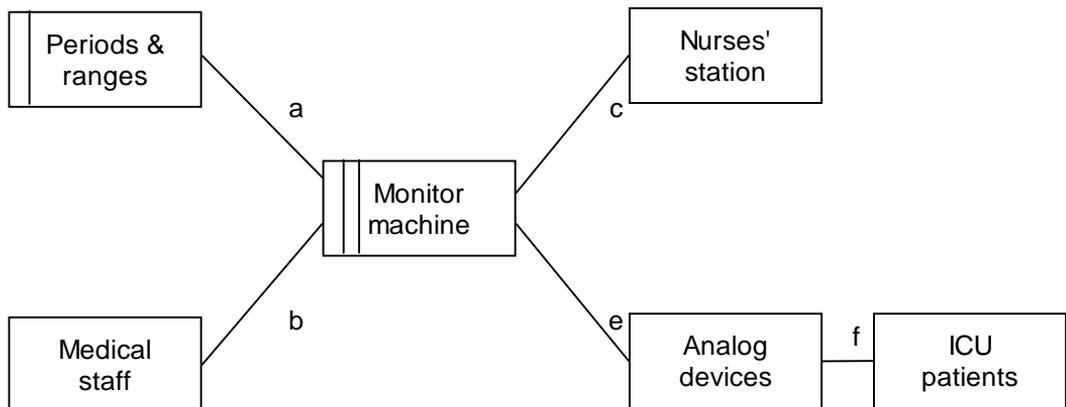
1.7 References

<List of other documents providing more detailed information>

2 OVERALL DESCRIPTION

2.1 Product Context

<Context diagram with Jackson (2001) notation>



a: Period, Range, PatientName, Factor
 b: EnterPeriod, EnterRange,
 EnterPatientName, EnterFactor

c: Notify
 e: RegisterValue
 f: FactorEvidence

Figure 1. Patient monitoring annotated context diagram

2.2 Product Perspective

2.2.1 System Interfaces

ID	Description	IS
S1		Y/N

<Examples of system interfaces>

Software: purchased components, reused components, common components

Hardware that must be supported by the software

Communications interfaces: LAN, WLAN, modem

2.2.2 User Interfaces

ID	Description	IS
----	-------------	----

I1		Y/N
----	--	-----

<Examples of user interfaces>

Dialogs: main, login, search, error, start up (splash), save, open, close, replace, send, print, print-preview, change password, change font, options, ...

2.2.3 Logical User Interface Characteristics

ID	Description	IS
L1		Y/N

<i>Required screen formats</i>
<i>Page or window layouts</i>
<i>Contents of any reports or menus</i>
<i>Availability of programmable function keys</i>

2.2.4 Operation Types

ID	Description	DR
O1		Y/N

<i>Regular</i>
<i>Degraded</i>
<i>Maintenance</i>
<i>Training</i>
<i>Emergency</i>
<i>Alternate-site</i>
<i>Peacetime</i>
<i>Wartime</i>
<i>Ground-based</i>
<i>Flight</i>
<i>Active</i>
<i>Idle</i>
<i>Backup</i>

2.3 Product Functions as Use Cases

ID	Name	UCD
UC1		Y/N

<i>Update weather forecast</i>
<i>Monitor untreated roads</i>
<i>Record treated roads</i>
<i>Produce de-icing schedule</i>
<i>Record truck changes</i>
<i>Record weather station readings</i>
<i>Amend de-icing schedule</i>
<i>Identify faulty weather station</i>
<i>Record new weather station</i>
<i>Record road</i>

2.4 Users

ID	User Type	Count	Skill Level	Use Pattern	Represented by	Priority
US 1						

<Examples of user types>

<i>Chemists</i>
<i>Buyers</i>
<i>Chemical stockroom staff</i>
<i>Health and safety staff</i>
<i>School children</i>
<i>Road engineers</i>
<i>Project managers</i>

2.5 Problem Domain Description

2.5.1 Important Domain Properties

ID	Description
DP1	

<Examples of domain properties, i.e. external factors that have an effect on the product, but are not mandated requirements constraints>

One ton of de-icing material will treat 3 miles of single lane roadway.

Drivers cannot work more than three hours without a break.

The existing application is 10,000 lines of C code.

2.5.2 Business Process Description

<E.g. UML activity diagram>

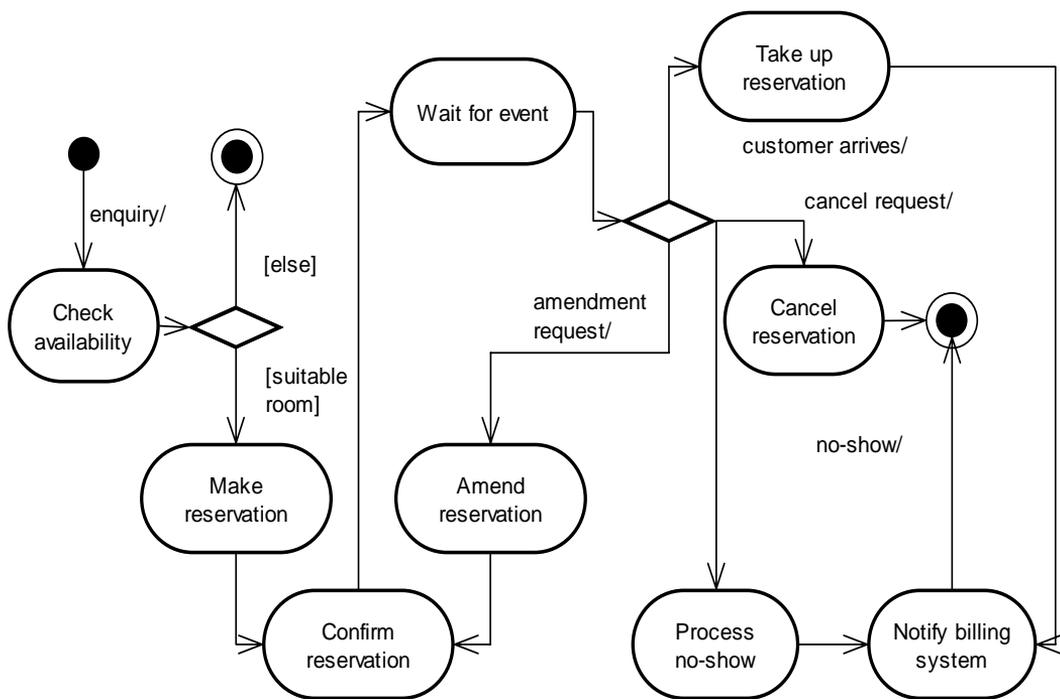


Figure 2. Business process for hotel reservation

2.6 System Requirements Allocations

2.6.1 Apportioning of Requirements

<Requirements criticalities and precedences>

<Tasks, features, or use cases that are implemented in initial version>

2.6.3 Manual Operations

<Operations that need to be done but are not supported by the software>

2.6.4 Restrictions, Limitations, and Exclusions

<Properties that have been allocated to a specific future release etc.>

2.7 Constraints

ID	Description	DR
C1		Y/N

<i>Regulatory policies</i>
<i>Hardware limitations</i>
<i>Parallel operation</i>
<i>Audit functions</i>
<i>Control functions</i>
<i>Higher-order language requirements</i>
<i>Signal handshake protocols</i>
<i>Criticality of the application</i>
<i>Current implementation environment</i>
<i>Partner applications</i>
<i>Commercial Off-The-Shelf (COTS) packages</i>
<i>Anticipated workplace environment</i>
<i>Schedule constraints</i>
<i>Financial constraints</i>
<i>Price constraints</i>
<i>Safety requirements</i>

2.8 Assumptions

ID	Description	DR
A1		Y/N

<Example assumptions, i.e. things that are not facts but could affect the requirements stated in RD>

<i>Plan of using commercial components (which)</i>
<i>Issues related to development or operating environment</i>

2.9 Dependencies

ID	Description	DR
D1		Y/N

<Example dependencies, if these are documented elsewhere, e.g. in project plan, refer to respective documents>

<i>Component x is developed by project y; x is on the critical path for this project</i>
<i>Roads that have been treated will not need treating for at least two hours. Road treatment will stop at county boundaries.</i>
<i>The Bureau's forecasts will be transmitted according to their specification 1003-7 issued by their engineering department.</i>

3 SPECIFIC REQUIREMENTS

3.1 Performance Requirements

ID	Performance Requirements	DR
P1		Y/N

<i>The product shall schedule such that the minimum necessary amounts of de-icing material are spread on roads.</i>
<i>The product shall schedule such that the rescheduled de-icing truck is estimated to arrive at the breakdown location within 30 minutes of breakdown notification.</i>

3.1.1 Response and Processing Times

ID	Response and Processing Times	DR
RP1		Y/N

<i>95% of catalog database queries shall be completed within 2 seconds on a 450 MHz Pentium II PC running Microsoft Windows 2000 with at least 50% of the system resources free. (Notice that having a standard PC defined in appendix you would not need to state it here anymore.)</i>
<i>95% of the transactions shall be processed in less than 1 s.</i>
<i>Scrolling one page up or down in a 200 page document shall take at most 1 s. Searching for a specific keyword shall take at most 5 s.</i>
<i>When moving to the next field, typing must be possible within 0.2 s. When switching to the next screen, typing must be possible within 1.3 s. Showing simple report screens must take less than 20 s. (Valid for 95% of the cases in standard load.)</i>

3.1.2 Precision

ID	Precision Requirements	DR
PR1		Y/N

<i>All the calculations will be done with 6 significant numbers.</i>
<i>All the user interface fields for currency will show 3 digits after decimal period.</i>
<i>The name field shall have 150 characters.</i>

3.1.3 Capacities

ID	Capacity Requirements	DR
CA1		Y/N

<i>How many users will be online simultaneously on average</i>
<i>How many queries will they likely run per day on average</i>
<i>How many users will be online simultaneously on maximum</i>
<i>How many queries will they likely run per day on maximum</i>
<i>The product shall use <16 MB of memory even if more is available.</i>

3.2 Data Requirements

3.2.1 Database

Database Vendor	
Version	
User Count, Normal	
User Count, Max	

Database Size	
---------------	--

3.2.2 Data Model

<E.g. UML class diagram with user visible classes only, including entities, attributes, and relations>

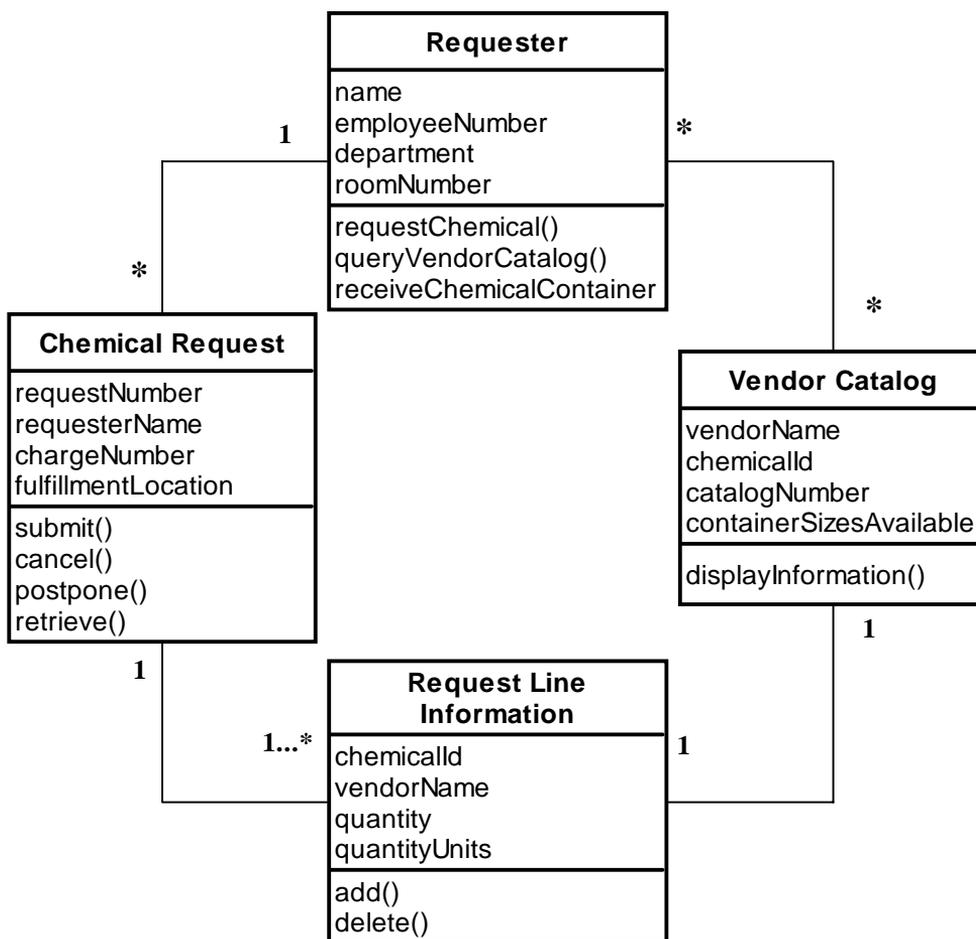


Figure 3. Class diagram for part of the Chemical Tracking System

3.3 Software System Attributes

3.3.1 Reliability

ID	Description	DR
----	-------------	----

RE1		Y/N
-----	--	-----

During the first week of operation at most 1 non-critical system failure is acceptable.

3.3.2 Availability

ID	Description	DR
AV1		Y/N

System must be available for use xx.xx % of the time

Mean time to repair after a failure is at most 1 hour

3.3.3 Security

ID	Security Requirements	DR
SE1		Y/N

Access to system must be controlled

Access to information must be controlled

Communications must be encrypted

Data back up and restore is required

Cryptographic operations must be done with algorithm x

Logs and history data sets are needed

Function x and y must be located in different modules

3.3.4 Maintainability

ID	Description	DR
MA1		Y/N

The supplier's hot-line shall analyze 95% of the problem reports within 2 work hours. Urgent defects (no work-around possible) shall be repaired within 30 work hours in 95% of the cases.

When repairing a defect, related non-repaired defects shall be less than 0.5 on average.

For a period of two year, the supplier shall enhance the product at a cost of ___ per Function Point.

3.3.5 Supportability

ID	Description	DR
SU1		Y/N

<i>The naming conventions in document x must be adhered to</i>
<i>Installation of a new version shall leave all database contents and personal settings unchanged.</i>
<i>Supplier shall station a qualified developer at the customer's site.</i>
<i>Supplier shall deposit code and full documentation of every release and correction at ____.</i>

3.3.6 Portability

ID	Description	DR
PO1		Y/N

<i>System must implemented using language x</i>
<i>At most x % of the code can be host dependent</i>
<i>At most x % of the components can be host dependent</i>
<i>A particular compiler or language subset must be used</i>
<i>A particular operating system must be used</i>

3.3.7 Usability

ID	Description	DR
UA1		Y/N

<i>A clerk typist grade 4 can do function X in Z min after 1 h of training</i>
<i>A normal user can do all daily tasks x after 4-hour training</i>
<i>A power user can learn all the daily tasks from the user guide in 4 hours</i>
<i>Novice users shall perform tasks Q and R in 15 minutes. Experienced users complete tasks Q, R, and S in 2 minutes.</i>
<i>80% of users shall find the system easy to learn. 60% shall recommend the system to others.</i>

3.3.8 Scalability

ID	Description	DR
----	-------------	----

SC1		Y/N
-----	--	-----

In the start up phase the system will have 10 users, but it must support up to 1000 users during the first year of operation.

3.4 Other Nonfunctional Requirements

<Any other non-functional requirements should be added here>

ID	Description	DR
NF1		Y/N

3.4.1 Documentation Requirements

ID	Document	Type	Audience
DO1			

<i>ID</i>	<i>Document</i>	<i>Type</i>	<i>Audience</i>
<i>DO1</i>	<i>User Manual</i>	<i>Printed/On-line</i>	<i>All user types</i>
<i>DO2</i>	<i>Installation Guide</i>	<i>Printed/On-line</i>	<i>System administrator</i>
<i>DO3</i>	<i>Configuration Guide</i>	<i>Printed/On-line</i>	<i>System administrator</i>
<i>DO4</i>	<i>Read Me file</i>	<i>On-line</i>	<i>System administrator</i>

3.4.2 Miscellaneous Requirements

<Any requirements that do not fit elsewhere>

ID	Type	Description	DR
R1			Y/N

3.4.3 Likely Changes

ID	Description	When
LC1		

Our investigation into whether or not the new version of the processor will be suitable for our application is not yet complete.

<i>The government are planning to change the rules about who is responsible for de-icing the motorways, but we do not know what the changes might be.</i>

3.4.4 Copyright, Legal, and Other Notices

ID	Description	DR
N1		Y/N

<i>Personal information must be implemented so as to comply with the Data Protection Act.</i>

<i>The product must confirm with the disabled access laws.</i>
--

3.5 Design Constraints

ID	Description	DR
DC1		Y/N

<i>Software must be written in Visual Basic</i>

<i>The application must run on both our new and old platforms.</i>
--

<i>Use the class library from Developer's Library 99-724 on the corporate IT server.</i>
--

<i>Compatibility with the legacy data base must be maintained.</i>
--

<i>Use our C++ coding standard.</i>

1 REQUIREMENTS DEVELOPMENT PROCESS

This appendix presents only the requirements development process part of the BaRE Guide.

1.1 Step 1: Establish objectives

RD topics	1.1 Customer problem 1.2 Business goals 1.3 Stakeholders 1.4 Product purpose 1.5 Product position statement
Participants	Initiator of the development effort Stakeholders
Techniques	Introspection Document reviews Electronic requirements Interviews Negotiation meetings
Tasks	1. Interview the initiator to get started with the task and to identify initial stakeholders 2. Review documents and introspect to get a good idea about the objectives 3. Interview stakeholders for deeper understanding of the task, use email when appropriate 4. Do web searches to help develop the product position statement 5. Arrange meetings with all stakeholders to negotiate about common understanding of the objectives

1.2 Step 2: Understand context

RD topics	2.1 Product context 2.2.1 System interfaces 2.2.2 User interfaces 2.2.3 Logical user interface characteristics 2.2.3 Operation types
-----------	--

	2.3 Product Functions as use cases 2.4 Users 2.5.1 Important domain properties 2.5.2 Business process description 2.7 Constraints – Initial
Participants	Stakeholders Users
Techniques	Introspection Document reviews Electronic requirements Interviews Diagrams: context, workflow
Tasks	1. Develop own understanding of the context, external interfaces, and the product in the defined context

1.3 Step 3: Organize knowledge

RD topics	All the previously developed RD topics except for 2.8 Constraints, i.e. 1.1-1.5 and 2.1-2.5
Participants	Stakeholders Users
Techniques	Meetings, negotiation or review
Tasks	1. Arrange meeting(s) to review named requirements document topics and to prioritize the business goals

1.4 Step 4: Elicit requirements

RD topics	3.1 Performance requirements – Initial 3.2 Data requirements 3.3 Software system attributes – Initial 3.4.1 Documentation requirements 3.4.2 Miscellaneous requirements – Initial App 2: Typical computer configuration App 2: Detailed requirements IS: 2. System interfaces IS: 3.x Dialog maps
-----------	---

	IS: 3.x User interfaces IS: 4. Refined data model IS: 5. Administrative users
Participants	Stakeholders (including technical experts) Identified users
Techniques	Introspection Document reviews Electronic requirements Structured interviews Meetings: brainstorming, workshops, negotiation Prototypes, horizontal/user interfaces Diagrams: data model, dialog maps
Tasks	1. Collect requirements from documents and individual people, use introspection 2. Broaden the requirements coverage with brainstorming session(s) 3. Deepen the requirements with use case workshop(s) 4. Narrow and prune the requirements, resolve conflicts with negotiation meetings

1.5 Step 5: Prepare for problems

RD topics	2.7 Constraints – Final 2.8 Assumptions 2.9 Dependencies 3.1 Performance requirements – Final 3.3 Software System attributes – Final 3.4.2 Miscellaneous requirements – Final 3.4.3 Likely changes 3.4.4 Copyright, legal, and other notices 3.5 Design constraints
Participants	Stakeholders (including technical experts) Users
Techniques	Introspection

	Electronic requirements Structured interviews Meetings: brainstorming, negotiation, workshops Checklists
Tasks	1. Try to uncover potential future problems with checklists 2. Verify availability of needed modules, components etc. with email, web searches, and interviews 3. Arrange brainstorming sessions to broaden and define requirement categories

1.6 Step 6: Prioritize requirements

RD topics	2.6.1 Apportioning of requirements 2.6.2 Scope of initial release 2.6.3 Manual operations 2.6.4 Restrictions, limitations and exclusions
Participants	Stakeholders Users
Techniques	Introspection Electronic requirements Interviews Meetings, negotiation or workshops
Tasks	1. Develop a requirements allocation plan 2. Review and revise the plan in meeting(s)

1.7 Step 7: Complete requirements document

RD topics	1.6 Document overview 1.7 References All other topics in the document
Participants	Stakeholders Users
Techniques	Reading Personal review
Tasks	1. Add references when you use them; at latest

	now 2. Validate the document personally 3. Complete missing sections
--	--

1.8 Step 8: Analyze requirements document

RD topics	IS: 6. CRUD-matrix
Participants	Stakeholders Users
Techniques	Prototypes, vertical/technical Requirements analysis checklists Develop CRUD-matrix Negotiation meetings
Tasks	1. Develop CRUD matrix from data model and use cases 2. Analyze individual requirements with checklists 3. Develop prototypes and review them with users 4. Negotiate with stakeholders about possible problems, possibly by organizing meetings

1.9 Step 9: Validate requirements document

RD topics	RD3-User Manual
Participants	Stakeholders Users
Techniques	Prototypes, horizontal Checklists Review
Tasks	1. Develop an initial user manual from the requirements document 2. Develop horizontal prototypes 3. Use checklists to validate requirements and the requirements document 4. Review the requirements document, user manual, and prototypes

RE Practice and Technique Summary

This appendix summarizes the good RE practices from the BaRE documentation that are not included in the BaRE method. Thus this listing provides ideas on how to extent the basic method for RE. The letters in the tables refer to the books that provide further information on the given practices. The letters refer to the books as follows:

- A – Lauesen, S. (2002). Software Requirements - Styles and Techniques. Harlow, Addison-Wesley.
- D – Davis, A. M. (1993). Software Requirements: Objects, States, and Functions, Prentice Hall.
- K – Kotonya, G. and I. Sommerville (1998). Requirements Engineering: Processes and Techniques. Chichester, John Wiley & Sons.
- L – Leffingwell, D. and D. Widrig (1999). Managing Software Requirements: A Unified Approach. Reading, Massachusetts, Addison-Wesley.
- S – Sommerville, I. and P. Sawyer (1997). Requirements Engineering: A Good Practice Guide. Chichester, England, John Wiley & Sons.
- W – Wiegers, K. E. (1999). Software Requirements. Redmond, Washington, Microsoft Press.

The Lauesen (A) and Davis (D) books are only referred to in Section 0 (Requirements Specification Techniques).

1. Good Practices for Elicitation

Practice	K	L	S	W
Write Vision and Scope				●
Select Product Champions				●
Establish Focus Groups				●
Define Quality Attributes				●
Examine Problem Reports				●
Be Sensitive to Organizational and Political Considerations			●	
Identify and Consult System Stakeholders			●	
Use Business Concerns to Drive Requirements Elicitation			●	
Look for Domain Constraints			●	
Record Requirements Rationale			●	
Collect Requirements From Multiple Viewpoints			●	
Use Scenarios to Elicit Requirements			●	
Define Operational Processes			●	

2. Good Practices for Documenting

Practice	K	L	S	W
Identify Sources of Requirements			●	●
Record Business Rules				●
Create Requirements Traceability Matrix				●
Supplement Natural Language with Other Descriptions of Requirements			●	
Specify Requirements Quantitatively			●	

3. Good Practices for Analysis

Practice	K	L	S	W
Analyze Feasibility			●	●
Prioritize Requirements			●	●
Model the Requirements				●
Create a Data Dictionary			●	●
Apply Quality Function Deployment				●
Provide Software to Support Negotiations			●	
Classify Requirements Using a Multi-Dimensional Approach			●	
Use Interaction Matrices to Find Conflicts and Overlaps	●		●	
Assess Requirements Risks			●	

4. Good Practices for Verification

Practice	K	L	S	W
Write Test Cases from Requirements	●			●
Define Acceptance Criteria				●

5. Good Practices for Validation

Practice	K	L	S	W
Check that the Requirements Document Meets Your Standards			●	
Organize Formal Requirements Inspections			●	●
Use Multi-Disciplinary Teams to Review Requirements			●	
Use Prototyping to Animate Requirements			●	

6. Good Practices for Requirements Management

Practice	K	L	S	W
Establish Change Control Board				●
Perform Change Impact Analysis				●
Trace Each Change to All Affected Work Products				●
Track Requirements Statuses				●
Measure Requirements Stability				●
Define Traceability Policies	●		●	
Maintain Traceability Manual	●		●	
Identify Global System Requirements	●		●	
Identify Volatile Requirements	●		●	
Configuration Management and Change Management		●		

7. Requirements Specification Techniques

Method	A	D	K	L	S	W
Pseudocode				●		
Finite State Machines		●		●		
Decision Trees and Decision Tables	●	●		●		
Graphical Decision Trees				●		
Activity Diagrams	●			●		
Entity-Relationship Models				●	●	●
Object-Oriented Modeling/Approaches			●	●		
Data Flow Diagrams	●	●	●	●	●	●
Semantic Data Models			●			
Formal Methods			●			
Tasks	●					
State Diagrams	●					
State Transition Diagrams	●				●	●
Class Diagrams	●				●	●
Collaboration Diagrams	●					
Sequence Diagrams, Events, Messages	●					
Requirements Engineering Validation System		●				
Petri Nets		●				
Program Design Language		●				
Requirements Language Processor		●				
Specification and Description Language		●				
PAISLey		●				
Process Model					●	

Requirements Changes in BaRE

Product Name	BaRE
Product Version	1.0
Document Created	2002-10-08
Document Modified	2003-08-07
Project Name	RETICE
Confidentiality	Confidential
Customer	tSoft
Preparers	un

Copyright © 2002 by Uolevi Nikula. Permission is granted to use, modify, and distribute this document internally. It may not be sold or used for commercial gain or other purposes without prior written permission.

This BaRE method document template version 1.0 has been developed as a part of the licentiate thesis of Uolevi Nikula, Lappeenranta University of Technology, Finland, in conjunction with the tSoft technology transfer program of the University of Joensuu, Finland. The template is currently under test use in tSoft projects and will be revised based on the experiences from industrial projects run under the tSoft program.

ChangID	Date	Issuer	Description	Status	CR
CH1	8.10.02	un	Add clearer "how to start" steps; adoption strategy		
CH3	8.10.02	ib	RDT: Add "Demo mode" to Operating Types		Y
CH4	8.10.02	ib	RDT: Section 3.1 should be only an empty header		Y
CH5	8.10.02	ib	App4: 2.1.2 Event/Use Case – remove Event since they are not discussed in BaRE		
CH6	9.10.02	am	Add example for CRUD-matrix use		
CH7	9.10.02	am	Add "RE in Short for Customer" guide		Y
CH8	9.10.02	am	Clarify the baseline/rqt dev vs. RM figure		
CH9	9.10.02	un	BaRE CM/RM process refinement: how to maintain changes, statuses; document vs. product versions with multiple doc baselines; as a list in RD binder,		
CH10	9.10.02	un	Add Enterprise Architect in tool suggestions		
CH11	9.10.02	un	Provide all templates and documents in Finnish		
CH12	17.10.02	un	RE in Short: Fig. 6&10 the same, change later to a reference		
CH13	17.10.02	un	Add empty templates with clear and full example documents to the package		
CH14	17.10.02	un	Clarify/emphasize more the iterative nature of software development, not only RE		
CH15	17.10.02	un	Training – clarify/refine, develop packages (e.g. use cases)		
CH16	17.10.02	un	Change "principal requirements engineer" to "RE coordinator"		
CH17	17.10.02	un	Consider adding "Connection Domain" to RDT		

CH18	4.11.02	un	Change "Evaluation" to "Review"		
CH19	4.11.02	un	Add Prioritization to techniques		
CH20	4.11.02	un	Introduce Change Control Board to change management, could consist of a single person		Y
CH21	4.11.02	un	Introduce Product Champion/key user role		
CH22	4.11.02	un	Consider using Application Domain instead of Problem Domain		
CH23	4.11.02	un	Consider adding Perspective Based Reading to techniques		
CH24	13.11.02	rn	Add testing to BaRE; rejected since already included in the analysis checklist		
CH25	13.2.03	un	Change 2.6 "System Requirements Allocation" to "Requirements Allocation"		
CH26	13.2.03	un	Change 2.6.1 "Apportioning of Requirements" to "General Principles"		
CH27	13.2.03	un	Suggest adding "2.6.5 Future Releases" in method adaptation		
CH28	13.2.03	un	Remove Restrictions from 2.6.4 heading, seems repetition with available knowledge in English		
CH29	29.1.03	jk	Replace "3 Specific Requirements" with plain "3 Requirements"		
CH30	14.2.03	un	Suggest 3 baselines: as-specified, as-implemented, and as-released		
CH31	5.3.03	un	Add "Unique Identification" to RM, not just versions		
CH32	5.3.03	un	Add "Use Requirement Templates" to RM techniques		
CH33	7.5.03	un	Move glossary to RDT and include IS, DR etc. in it - as examples & actual explanation of the abbreviations; suggest creating glossary as an appendix		

			as an alternative to the included one		
CH34	7.5.03	un	Add example SW system requirements to RDT		
CH35	11.5.03	un	CMM Chance Request template data: Product, Tracking Number, submitter information (Organization Name, Contact Name, Telephone, Mailing Address), Submission Date, Short Title, Change Location Tag, Proposed Change, Rationale for Change.		
CH36	23.4.03	al	Add more introduction to RD: why these topics, customer role, motivate to read/validate, for customer what & why – topics and structure		
CH37	19.3.03	un	Define checklist for known pitfalls in adoption based in done case studies		
CH38	1.6.03	un	Suggest minimum RE to anybody, i.e. document business goals, stakeholders, use case names, and requirements as identifier and short name		
CH39	2.6.03	un	Define standard training modules for BaRE and RE		
CH40	5.6.03	un	Add introduction to documentation (JK, JH): a 3-4 page abstract of what the RD is, how to read it, and how to start working with it		
CH41	5.6.03	un	Develop a proper full example RD as an example		
CH42	5.6.03	un/al	Add more explanatory text in the RDT and other documents		
CH43	27.11.02	ijh	Prioritizing forgotten; add MoSCoW and use case ranking		

Change Requests for BaRE

Product Name	BaRE
Product Version	1.0
Document Created	2003-06-25
Document Modified	2003-08-07
Project Name	RETICE
Confidentiality	Confidential
Customer	tSoft
Preparers	un

Copyright © 2002 by Uolevi Nikula. Permission is granted to use, modify, and distribute this document internally. It may not be sold or used for commercial gain or other purposes without prior written permission.

This BaRE method document template version 1.0 has been developed as a part of the licentiate thesis of Uolevi Nikula, Lappeenranta University of Technology, Finland, in conjunction with the tSoft technology transfer program of the University of Joensuu, Finland. The template is currently under test use in tSoft projects and will be revised based on the experiences from industrial projects run under the tSoft program.

TABLE OF CONTENTS

Table of Contents	1
1 Change Requests	2
1.1 Add Demo Mode – CH3	2
1.1.1 Description	2
1.2 Performance Requirement Inconsistent – CH4	2
1.2.1 Description	2
1.3 RE in Short for Customer Guide – CH7	2
1.3.1 Description	2
1.4 Introduce CCB – CH20	2
1.4.1 Description	2
1.4.2 Rationale	2
1.4.3 History	3

1 CHANGE REQUESTS

1.1 Add Demo Mode – CH3

1.1.1 Description

Discussion with IB about the operating modes indicated that a realistic mode in their applications could be a demo mode which is common for software sold in the Internet or on CD.

1.2 Performance Requirement Inconsistent – CH4

1.2.1 Description

Section 3.1 in the RD template is a group header level for multiple requirements. The adopted convention is that group headers are empty, so place the performance requirements elsewhere.

1.3 RE in Short for Customer Guide – CH7

1.3.1 Description

In bespoke software development, customer needs to understand basics about RE, but not as much as the developers. Develop another guide that is targeted for the customers.

1.4 Introduce CCB – CH20

1.4.1 Description

Introduce Change Control Board that has the task to regularly check change requests and act on them promptly.

1.4.2 Rationale

If there is no one with named responsibility to check the requirements, the performance is unpredictable. Having a CCB named to do it allows for flexibility – it can consist of one person or representatives from many stakeholder groups (customers, development, sales, management, etc.).

1.4.3 History

Problem identified when observing two case studies. One that had weekly development meetings seemed to address changes to their product promptly, but another one that did not check the change requests regularly appeared to have some problems that could be caused by this.

ACTA UNIVERSITATIS LAPPEENRANTAENSIS

143. PUUMALAINEN, KAISU. Global diffusion of innovations in telecommunications: Effects of data aggregation and market environment. 2002. 153 s. Diss.
144. SARRETTE, CHRISTINE. Effect of noncondensable gases on circulation of primary coolant in nuclear power plants in abnormal situations. 2003. 114 s. Diss.
145. SAARENKETO, SAMI. Born globals – internationalization of small and medium-sized knowledge-intensive firms. 2002. 247 s. Diss.
146. IKONEN, KIRSI. Metal surface and subsurface inspection using nondestructive optical methods. 2002. U.s. Diss.
147. OLLIKAINEN, Mikael. Origins of production errors and significance of employee empowerment in reducing production error amount in sheet metal fabricating industry. 2003. 153 s. Diss.
148. KOUVO, PETRI. Formation and control of trace metal emissions in co-firing of biomass, peat and wastes in fluidised bed combustors. 2003. U.s. Diss.
149. MOOSAVI, ALI. Transport properties of multi-phase composite materials. 2003. U.s. Diss.
150. SMOLANDER, KARI. On the role of architecture in systems development. 2003. U.s. Diss.
151. VERENICH, SVETLANA. Wet oxidation of concentrated wastewaters: process combination and reaction kinetic modeling. 2003. U.s. Diss.
152. STÅHLE, PIRJO, STÅHLE, STEN & PÖYHÖNEN, AINO. Analyzing dynamic intellectual capital: System-based theory and application. 2003. 191 s.
153. HAATAJA, JORMA. A comparative performance study of four-pole induction motors and synchronous reluctance motors in variable speed drives. 2003. 135 s. Diss.
154. KURRONEN, PANU. Torque vibration model of axial-flux surface-mounted permanent magnet synchronous machine. 2003. 123 s. Diss.
156. KUIVALAINEN, OLLI. Knowledge-based view of internationalisation – studies on small and medium-sized information and communication technology firms. 2003. U.s. Diss.
157. AHOLA, JERO. Applicability of power-line communications to data transfer of on-line condition monitoring of electrical drives. 2003. 141 s. Diss.
158. 1st Workshop on Applications of Wireless Communications. Edited by Jari Porras, Jouni Ikonen and Pekka Jäppinen. 2003. 67 s.
159. TUUTTI, VEIKKO. Ikkunoiden lasirakenteet ja niiden valintaan vaikuttavat tekijät. 2003. 294 s. Diss.
160. METSÄMUURONEN, SARI. Critical flux and fouling in ultrafiltration of proteins. 2003. U.s. Diss.
161. TUIMALA, JARNO. Aiding the strategic purchasing benchmarking process by decision support systems. 2003. U.s. Diss.
162. SORSA, TIA. The interaction of the calcium sensitiser levosimendan with cardiac troponin C. 2003. U.s. Diss.
163. KOVANEN, JANNE. Improving dynamic characteristics of open-loop controlled log crane. 2003. 97 s. Diss.
164. KURTTILA, HEIKKI. Isentropic exergy and pressure of the shock wave caused by the explosion of a pressure vessel. 2003. 114 s., liitt. Diss.

165. KÄMÄRÄINEN, JONI-KRISTIAN. Feature extraction using Gabor filters. 2003. U.s. Diss.
166. ZAMANKHAN, PARSAN. Complex flow dynamics in dense granular flows. 2004. U. s. Diss.
167. MIELIKÄINEN, JARNO. Lossless compression of hyperspectral images. 2003. U.s. Diss.
168. LI, XIAOYAN. Effect of mechanical and geometric mismatching on fatigue and damage of welded joints. 2003. U.s. Diss.
169. OJANEN, VILLE. R&D performance analysis: case studies on the challenges and promotion of the evaluation and measurement of R&D. 2003. U.s. Diss.
170. PÖLLÄNEN, RIKU. Converter-flux-based current control of voltage source PWM rectifiers – analysis and implementation. 2003. 165 s. Diss.
171. FRANK, LAURI. Mobile communications within the European Union: the role of location in the evolution and forecasting of the diffusion process. 2003. U.s. Diss.
172. KOISTINEN, PETRI. Development and use of organizational memory in close and long-term cooperation between organizations. 2003. 170 s. Diss.
173. HALLIKAS, JUKKA. Managing risk in supplier networks: case studies in inter-firm collaboration. 2003. U.s. Diss.
174. LINDH, TUOMO. On the condition monitoring of induction machines. 2003. 146 s. Diss.
175. NIKKANEN, MARKKU. Railcarrier in intermodal freight transportation network. 2003. 217 s. Diss.
176. HUISKONEN, JANNE. Supply chain integration: studies on linking customer responsiveness and operational efficiency in logistics policy planning. 2004. 151 s. Diss.
177. KUISMA, MIKKO. Minimizing conducted RF-emissions in switch mode power supplies using spread-spectrum techniques. 2004. 190 s. Diss.
178. SOPANEN, JUSSI. Studies of rotor dynamics using a multibody simulation approach. 2004. 91 s. Diss.
179. On the edge of fuzziness. Studies in honor of Jorma K. Mattila on his sixtieth birthday. Editors Vesa A. Niskanen and Jari Kortelainen. 2004. 132 s.
180. VÄISÄNEN, PASI. Characterisation of clean and fouled polymeric membrane materials. 2004. U.s. Diss.
181. IKÄVALKO, MINNA. Pas de deux of art and business: a study of commitment in art sponsorship relationships. 2004. 277 s. Diss.
182. ENQVIST, YUKO. Comprehensive study of crystal growth from solution. 2004. U.s. Diss.
183. JÄPPINEN, PEKKA. ME – mobile electronic personality. 2004. U.s. Diss.
184. HALME, TAPANI. Novel techniques and applications in generalised beam theory. 2004. 101 s. Diss.
185. LOISA, ANTTI. Studies on integrating kinematic design method with mechanical systems simulation techniques. 143 s., liitt. Diss.
186. 2nd Workshop on Applications of Wireless Communications. 2004. 74 s.
187. LI, XIAONING. Conflict-based method for conceptual process synthesis. 2004. U. s. Diss.