Knowledge Management in Software Process Improvement

Finn Olav Bjørnson

Doctoral Thesis

Submitted for the Partial Fulfilment of the Requirements for the Degree of

philosophiae doctor

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology

October 2007

ISBN printed - 978-82-471-3330-9

ISBN electronic - 978-82-471-3344-6

ISSN – 1503-8181

Doctoral theses at NTNU, 2007:152

Printed in Norway by NTNU Trykk, Trondheim

Abstract

Reports of software a development projects that miss schedule, exceeds budget and deliver products with poor quality are abundant in the literature. Both researchers and the industry are seeking methods to counter these trends and improve software quality.

Software Process Improvement is a systematic approach to improve the capabilities and performance of software organizations. One basic idea is to assess the organizations' current practice and improve their software process on the basis of the competencies and experiences of the practitioners working in the organization. A major challenge is to create strategies and mechanisms for managing relevant and updated knowledge about software development and maintenance. Insights from the field of knowledge management are therefore potentially useful in software process improvement efforts to facilitate the creation, modification, and sharing of software processes in any organization.

In the work presented in this thesis, we have made an overview of empirical studies on the effect of knowledge management in software engineering. We have categorized these studies according to a framework and we report findings on the major concepts that have been investigated empirically, as well as the research methods applied within the field. We have also investigated two main strategies for knowledge management, codification and personalization, through the application of four concrete methods in a software process improvement setting: Mentoring, Rational Unified Process, Process Workshops and Post Mortem Analysis.

We have classified the work in this thesis within three main themes:

RT1: Previous research on knowledge management in software engineering.
RT2: Application of knowledge management to improve the software process through codification of knowledge.
RT3: Application of knowledge management to improve the software process through personalization of knowledge.

The main contributions are:

C1: An overview of the research literature on empirical studies of knowledge management in software engineering.
C2: A method for tailoring the Rational Unified Process to the development process of a software consulting company.
C3: Improvements of the Process Workshops method by contextualization.
C4: Improvement of the root-cause analysis phase of the lightweight Post Mortem Analysis for more effective project retrospectives.
C5: Proposed methods to increase the learning effect of mentor programs in software engineering.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for partial fulfilment of the requirements for the degree of Philosophiae Doctor.

The work referred to has been performed at the Department of Computer and Information Science, NTNU, Trondheim, under the supervision of Professor Reidar Conradi.

This work was conducted as a part of the Software Process Improvement through Knowledge and Experience (SPIKE) research project, supported by the Research Council of Norway through grant 156701/220.

75% of the PhD scholarship was funded through SPIKE, the remaining 25% was funded by one year of mandatory teaching assistant duties at the Department of Computer and Information Science. The scholarship ran from 2003 to 2007.

Acknowledgements

First of all I would like to thank my advisor Reidar Conradi for his continuous support and advice during my PhD study, and his comments on the papers and drafts of this thesis. I would also like to extend a special thanks to my second advisor Torgeir Dingsøyr, first of all for inspiring me to take the knowledge management perspective to software process improvement, but also for our joint research and countless discussions and advice during my study, I couldn't have done it without you!

I would also like to thank, professor Tor Stålhane for our joint research towards one of our case companies, and Geir Kjetil Hanssen and Hans Westerheim for the joint research on RUP. The case companies who shall remain anonymous deserves a thanks for letting us study their improvement initiatives. Also thanks to Alf Inge Wang for allowing me to experiment on his students, Erik Arisholm for helping us analyze the data from said students, and the students for participating in our experiment. Finally a thanks to Jingyue Li for letting me participate in his research when I was still struggling with making a direction for my own research, and for lending me his template for the PhD thesis. I told you, you'd get a mention for it! There are many more that deserves a mention, so I'll extend a general thanks to the software engineering group at IDI, and the software engineering group at SINTEF, thank you for providing a stimulating environment with many interesting discussions. I'm also grateful to the SPIKE project and the research council of Norway for funding me through these years!

My family also deserves a special mention. Thanks to Ellen, Georg and Jon Harald for your support and encouragement through these years!

Also thanks to my friends in NTNUI-Dans, dancing the night away is a good way to relax after a hard day of research! And finally thanks to "rollespillgjengen", you know who you are, for instituting a PhD requirement on membership. Two down, how many more to go now? You provided an insane environment to get away from it all at the darkest times of paper and thesis writing.

Contents

List of Figures

List of Tables

Abbreviations

AR	Action Research
CMM	Capability Maturity Model
CoP	Community of Practice
COTS	Commercial-Off-The-Shelf
ICT	Information and Communication Technology
ISO	International Standards Organization
KM	Knowledge Management
LSO	Learning Software Organization
NTNU	Norwegian University of Science and Technology
OO	Object-Oriented
OTS	Off-The-Shelf
PEP	Process Engineering Process
PMA	Post Mortem Analysis
PWS	Project Workshops
QIP	Quality Improvement Program
RUP	Rational Unified Process
SE	Software Engineering
SEI	the Software Engineering Institute (SEI) at Carnegie Mellon University
SPI	Software Process Improvement
SR	Systematic Review
TQM	Total Quality Management
UP	Unified Process

1 Introduction

This chapter outlines the context and motivation for the research presented in this thesis. Research questions and claimed contributions are briefly presented together with a list of the papers. Finally, a thesis outline is presented.

1.1 Problem Outline

The research literature is filled with reports of software projects missing their schedules, exceeding their budgets, deliveries of software with poor quality and in some cases even wrong functionality. Both researchers and the software industry are seeking methods counter these trends and improve productivity and software quality. One approach to building better software products is software process improvement.

The fundamental belief of software process improvement is that improving the process will lead to improvements in the final product. A basic idea is to assess the organizations' current practice and improve their software process on the basis of the competencies and experiences of the practitioners working in the organization.

Since an organization's software development practices are ultimately based on the knowledge and competencies of its software developers and managers, Mathiassen et al. (2001) argue that software process improvement efforts depend on the implicit, individual knowledge of practitioners in an organization. To change software developers practices, the organization should improve the practitioners' existing knowledge (both theoretical and practical) of its software practices. In other words, knowledge about the new processes should be made available on different organizational levels. A major challenge for software process improvement initiatives is hence to create strategies and mechanisms for managing knowledge about software development. Insights from the field of knowledge management are therefore potentially useful in software process improvement efforts to facilitate the creation, modification, and sharing of software processes in an organization.

Lyytinen and Robey (1999) speaks of a learning failure in the software industry. Not only do many companies fail to learn and improve from previous experience, in time they have also learned to expect to fail. Over time many companies have come to expect and accept poor performance while creating organizational myths that perpetuates short-term optimization. Lyytinen suggest using knowledge management as a way to increase organizational intelligence in order to overcome these problems.

In an introductory chapter in the book Managing Software Engineering Knowledge, Edwards (2003) motivates the need for knowledge management in software engineering. He identifies six principal challenges, three categories of solutions, and two overall strategies that can be employed. The three types of solutions are technological-, people-, and process-solutions. The two overall strategies are the *codification* and *personalization* strategies, suggested by Hansen et al. (1999).

In (Wickert, 2001) the authors examine challenges facing small businesses when implementing knowledge management efforts. Small businesses are particularly vulnerable to knowledge erosion, yet they seldom have the time and resources needed to implement the knowledge management programs described for larger companies. However, the authors suggest that small businesses can benefit just as much from well thought out knowledge management efforts.

In the following thesis we have investigated how knowledge management can be used to help small and medium sized software companies improve their software processes.

1.2 Research Context

The work in this thesis has been carried out as part of the SPIKE project. SPIKE, Software Process Improvement based on Knowledge and Experience, is a R&D project in software process improvement (SPI) and software quality running in 2003 - 2005. The main contractor was Abelia, the leading interest group for knowledge and technology based companies in Norway. The research partners were: SINTEF, the Norwegian University of Science and Technology, and the University of Oslo/Simula Research Laboratory. The Information and Communications Technology (ICT) industry was represented through 10 Norwegian companies. The industrial partners were interested in improving their software projects, and were seeking better and concrete processes and methods that would help them deliver high quality software faster and cheaper.

The main goal of the SPIKE project was to define improved methods to increase competitive power and add value to Norwegian ICT businesses. Important factors were knowledge, competence, cooperation in networks, and learning and innovation, both national and international. More specifically this would be achieved through:

- Empirical studies to assess the results of revised or novel methods and techniques in industrial software projects.
- Common projects across companies to harvest, refine and reuse experiences and knowledge. Innovation of new knowledge on methods and techniques and the interaction between technology, organization and market.

- Dissemination of results of acquired knowledge and experience.
- Active participation in national and international fora to gather and spread knowledge and experience.
- PhD scholarships linked to the participating universities.

SPIKE continued effort on similar themes in its predecessors the PROFIT project, in 2000-2002, and the SPIQ project in 1997-1999. In total, more than 70 individual industrial studies in 30 different companies have been carried out through these projects. The results from the SPIKE project which ran to completion during the work in this thesis, was published as a book in (Conradi, 2006). The EVISOFT follow-up project to SPIKE, has already started and has been granted funding in 2006 to 2010.

1.3 Research Questions

The overall perspective for all studies carried out as part of this thesis was:

How can Knowledge Management (KM) be applied to Software Engineering (SE) in order to foster Software Process Improvement (SPI)?

In order to go from our overall topic and goal to specific studies and research questions, we have formulated the following questions:

What are we studying?
- We studied different approaches to software process improvement (in particular how new process knowledge is created and spread throughout an organization) from a knowledge management perspective in small and medium sized software companies.

Why are we interested in it?
- Because an organization's software practices are ultimately based on the knowledge and skills of their software developers, we wanted to find out how general theories on knowledge creation and knowledge transfer could be applied in the SE domain. Particularly in an SPI setting, where the key challenge is to change practices.

Why would this be of any interest to anyone else?
- This benefits both the research community who gains deeper insight in how the general KM theory applies to a specialized setting, and practitioners who gains insight into how they can improve SPI initiatives by actively applying KM theories.

The thesis presents five studies where software process improvement initiatives were studied from a knowledge management perspective. The cooperative nature of the research meant that the companies involved had a large impact of what technologies and methods we studied. Based on the researchers perspective on knowledge management and the companies needs to improve their processes, we agreed on four specific methods that both satisfied the researchers' goals and the goals of the industrial partners: Mentoring, tailoring of the Rational Unified Process (RUP), Process Workshops (PWS) and Post Mortem Analysis (PMA). We have formulated research questions that explore each of these technologies and have grouped them into three main themes for this thesis. Our first research theme concerns mapping out the field and investigating what has previously been done, our second and third research theme relates to the two major strategies for managing knowledge mentioned in section 1.1. We present the overall themes and questions briefly in this section, we give a rationale for choosing each question in chapter 3.

RT1: *Previous research on knowledge management in software engineering.*

- RQ1.1: *What concepts have been investigated empirically within the field of knowledge management in software engineering?*
- RQ1.2: *What are the research methods used in studying knowledge management in software engineering?*

RT2: *Application of knowledge management to improve the software process through codification of knowledge.*

- *RQ2.1: What do developers want from a knowledge sharing tool?*
- *RQ2.2: What are the limitations, benefits, prerequisites and cost of tailoring and introducing the Rational Unified Process?*
- *RQ2.3: How do available information, company context and goals affect the result of process workshops?.*

RT3: *Application of knowledge management to improve the software process through personalization of knowledge.*

- *RQ3.1: How can knowledge transfer through a mentor program be improved?*
- *RQ3.2: How do available information, company context and goals affect the learning effects during execution of process workshops?*
- *RQ3.3: How can sharing of project experience through project retrospectives be improved?*

1.4 Research Design

The study of software engineering has always been complex; the complexity arises from both technical issues, human issues in development and the interface between humans and systems. As the field of software engineering matures, there is an increased demand for empirically validated results not just the concept analysis and proof of concepts, which seems to have dominated the field so far (Glass, 2004). A recent trend in software engineering is an increased focus on empirical and Evidence-Based Software Engineering, EBSE (Dybå, 2005). The SPIKE project, which the work of this thesis was carried out in, also placed the demand that our results should be based on empirical studies and observations.

Empirical studies may be performed quantitatively, qualitatively or in combination. The choice of approach affects data collection, data analysis and threats to validity. Comparing quantitative and qualitative research, it could be argued that human behavior is one of the few phenomena that warrant a qualitative method.

Generally it could be argued that quantitative methods and statistics only show a correlation between a treatment and an outcome. In other words they only describe *what* happens in a specific case or a specific context. In order to map these results to a cause and effect construct you need a certain degree of qualitative methods to understand *why* the two items correlate.

Given our setting within the SPIKE project, where tight cooperation with the participating companies were both expected and ensured, the often short term goals of the companies, and our focus on the more human aspects of knowledge, we decided that a qualitative approach was more suited than a quantitative approach. Through close cooperation over an extended period with the participating companies we were able to get a good picture of how the companies were working. The positive side of this was that we get a very good insight into how our companies were functioning. The trade-off we had to make for this was the lack of generalizability to other companies. We tried to remedy this by comparing our results to the results of studies in other SPIKE companies and other studies in the literature, and we tried to explain our results through theoretical frameworks.

Figure 1 shows the studies performed and their relations to papers and contributions. The papers are listed in section 1.5 and the contributions are described in section 1.6.

As the figure shows, the thesis is built around five main studies. An early study of reuse of COTS is not included in the thesis, since it is not inside our final scope. Our contribution in study 0 was mainly data collection. Table 1 provides an overview of the technologies, contexts and research methods applied in the different studies.

Table 1: Overview of Studies

Study	**Focus**	**Research Method**	**Context**
Study 0	COTS	Survey with focused interviews	13 Norwegian ICT companies
Study 1	Mentoring	Action Research	Medium software consulting company
Study 2	RUP	Action Research	Medium sized software consulting company
Study 3	Process Workshops	Action Research	Small sized software consulting company
Study 4	Post Mortem Analysis	Controlled Experiment	142 4th year master students
Study 5	KM in SE	Systematic Review	Main electronic databases of the SE field

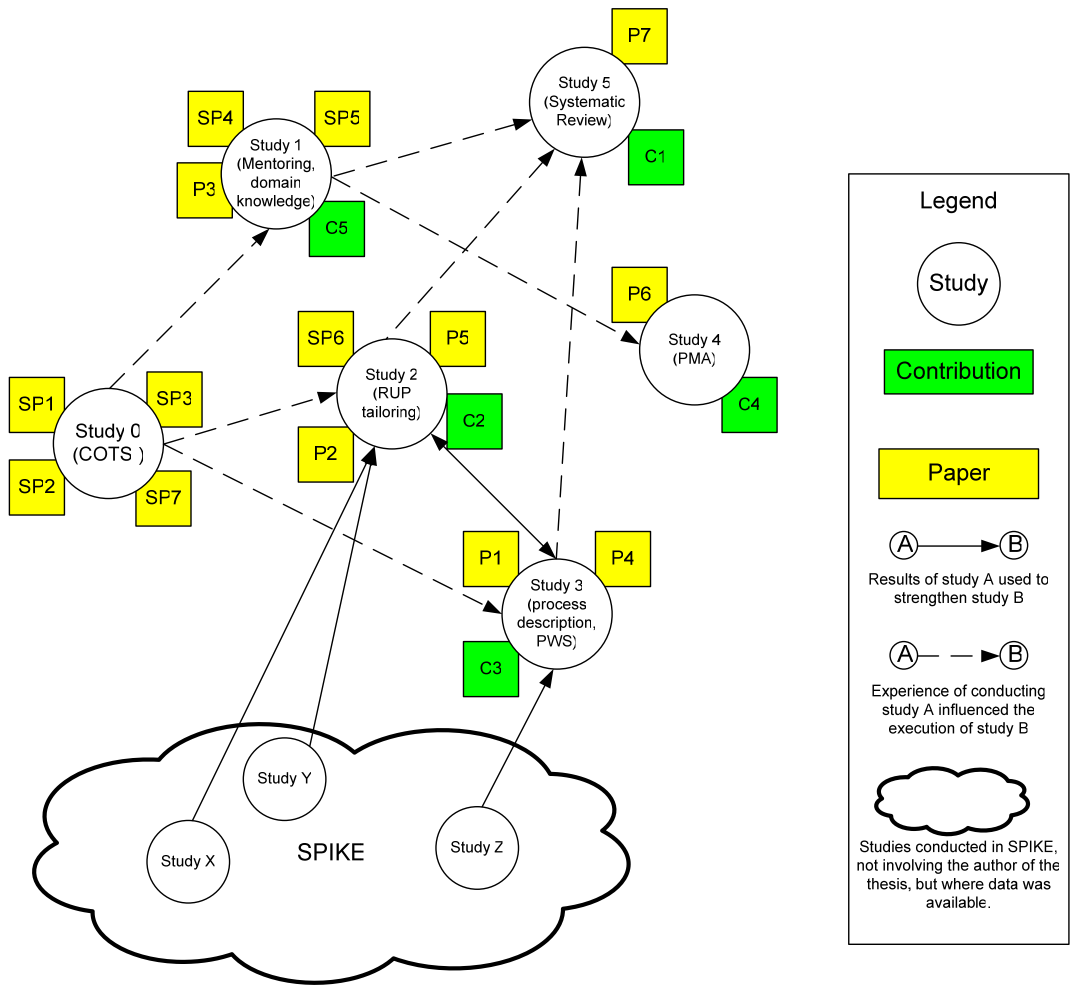
P7
Study 5 (Systematic Review)
C1
SP4
SP5
Study 1 (Mentoring, domain knowledge)
P3
C5
P6
Study 4 (PMA)
C4
SP6
P5
Study 2 (RUP tailoring)
C2
P2
SP1
SP3
Study 0 (COTS)
SP2
SP7
P1
P4
Study 3 (process description, PWS)
C3
Study Y
Study X
SPIKE
Study Z
Legend
Study
Contribution
Paper
Results of study A used to strengthen study B
Experience of conducting study A influenced the execution of study B
Studies conducted in SPIKE, not involving the author of the thesis, but where data was available.

Figure 1: Studies and their contribution

1.5 Papers

This section gives a short summary of the 7 papers included in this thesis. Together they describe the five main studies we build our results on. We briefly describe their relevance to the thesis and identify my contribution. The full papers can be found in Appendix A. In addition to describing the 7 included papers, we include the bibliography of 7 other papers also produced during the work on this thesis. The abstract of these papers can be found in appendix B. The papers included in the thesis have the designation P#, the secondary papers which are not included are designated SP#.

P1 Finn Olav Bjørnson and Tor Stålhane: "*Harvesting Knowledge through a Method Framework in an Electronic Process Guide*", Proc. 7th International Workshop on Learning Software Organizations (LSO), Kaiserslautern,

Germany, 2005, 107-111 (Post conference proceedings printed in Springer LNAI 3782, 2005, 86-90)
Relevance to this thesis: This paper presents our initial findings in study 3, and details how they envisioned their knowledge sharing project. It describes a tool based on the preferences of the developers and input from the research literature. The paper answers research question RQ2.1 and contributes towards contribution C3 and to some degree C2. The study contributes to a small degree towards research theme RT2.
My contribution: This paper is the result of a cooperation in SPIKE. I performed half of the interviews during the data gathering and was responsible for performing the analysis of the qualitative data. I was the leading author of this paper.

P2 Geir K. Hanssen, Hans Westerheim, and Finn Olav Bjørnson: "*Tailoring RUP to a defined project type: A case study*", Proc. 6th International Conference on Product Focused Software Process Improvement, Oulo, Finland, Springer LNCS 3547, 2005, 314-327
Relevance to this thesis: This paper presents our initial findings from study 2, it details the work with selecting a tailoring strategy for the Rational Unified Process and the work done in order to arrive at a downscaled version which was presented in a wiki web. The paper answers research question RQ2.2 and contributes towards C2. The study contributes to some degree towards research theme RT2.
My contribution: This work is the result of a cooperation in SPIKE. Two researchers were already involved with the company when I joined. I participated in the data gathering and analysis of the project, I was also involved with creating the workshop strategy to define their process framework. In addition I was heavily involved with choosing the research strategy.

P3 Finn Olav Bjørnson and Torgeir Dingsøyr: "*A study of a Mentoring Program for Knowledge Transfer in a Small Software Company*", Proc. 6th International Conference on Product Focused Software Process Improvement, Oulo, Finland, Springer LNCS 3547, 2005, 245-256
Relevance to this thesis: This paper presents findings from study 1. It describes our work to improve their mentor program of a company based on input from the developers and the research literature. It answers research question RQ3.1 and is the foundation of contribution C5. The study contributes to some degree towards research theme RT3.
My contribution: This work is the result of a cooperation in SPIKE. The workload for data collection and company meetings was shared equally between me and the coauthor. In addition I performed the literature survey of mentoring in organizational science, performed the majority of the analysis of the qualitative data, and I was the leading author of this paper.

P4 Finn Olav Bjørnson, Tor Stålhane, Nils Brede Moe, and Torgeir Dingsøyr: "*Defining Software Processes Through Process Workshops: A Multicase Study*", Proc. Of the 8th International Conference on Product Focused Software

Development and Process Improvement (PROFES'2007), LNCS 4589, Springer Verlag, 2007, 132-146.
Relevance to this thesis: This paper presents the application of a method for defining software processes, called the Process Workshops method. The results from applying the method in two different contexts are reported and discussed to provide contextualization for the method. The paper answers research question RQ2.3 and RQ3.2 and is the foundation of contribution C3. The study contributes to some degree to theme RT2 and in a large degree to research theme RT3.
My contribution: This work is the result of a large cooperation in SPIKE. One case was observed by me and another researcher (study 3 in Figure 1) the other case was observed by two other SPIKE researchers (Study Z in Figure 1). In addition to being heavily involved with one of the cases I was the leading author of this paper and coordinated the analysis and writing between the two groups.

P5 Geir Kjetil Hanssen, Finn Olav Bjørnson and Hans Westerheim: “*Tailoring and introduction of the Rational Unified Process”*, Proc. of EuroSPI 2007, LNCS 4764, Springer Verlag, 2007, 7-18.
Relevance to this thesis: This paper extends the results of paper P2 by combining it with two other SPIKE cases (Study X and Y in Figure 1) and a literature study. It answers research question RQ2.3 and contributes towards C2. The study is a major contribution to research theme RT2.
My contribution: This work is the result of a large cooperation in SPIKE. I was involved with one of the three main case studies, where I performed the majority of data collection and analysis. I was also responsible for about half of the literature study and conducting an analysis across all the cases and the literature.

P6 Finn Olav Bjørnson, Alf Inge Wang and Erik Arisholm: “*Improving the Effectiveness of Root Cause Analysis in a Retrospective Method: a Controlled Experiment*” Submitted Journal of Information and Software Technology.25p
Relevance to this thesis: This paper presents our findings from study 4, it outlines improvements to a retrospective method for eliciting project experience from software developers. The improvements was tested and validated in a controlled experiment. The paper answers research question RQ3.3 and is the foundation for contribution C4. The study is a major contribution to research theme RT3.
My contribution: I proposed the changes to the method and planned the experiment. I participated in the experiment as one of the lecturers and observed the implementation of it. I was also responsible for the qualitative analysis and was the leading author of the paper.

P7 Finn Olav Bjørnson and Torgeir Dingsøyr: “*Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts and Research Methods Used”* Submitted Journal of Information and Software Technology. 35p.
Relevance to this thesis: This paper presents our findings from study 5: an overview of empirical evidence in the field of knowledge management in

software engineering. It gives the answer to research questions RQ1.1 and RQ1.2 and is the foundation for contribution C1. The study is the major contribution to research theme RT1.
My contribution: This paper is one of the main contributions towards this thesis with me as leading author. I was responsible for conducting the initial search and exclusions. After we had narrowed our search down to the major papers, I conducted the synthesis of 2/3 of the selected papers.

The remaining papers were seen to be outside the scope of this thesis, so we will not go into details of their relevance and my contribution. The abstracts of the papers are included in appendix B.

SP1 Jingyue Li, Finn Olav Bjørnson, and Reidar Conradi: "*Empirical Study on COTS Component Classification*", Proc. International Workshop on COTS Terminology and Concepts, Redondo Beach, USA, 2004, 4p.

SP2 Jingyue Li, Finn Olav Bjørnson, Reidar Conradi, and Vigdis By Kampenes: "*An Empirical Study of COTS Component Selection Processes in Norwegian IT companies*", Proc. Of the International Workshop on Models and Processes for the Evaluation of COTS Components (MPEC), Edinburgh, Scotland, 2004, 27-30

SP3 Jingyue Li, Finn Olav Bjørnson, Reidar Conradi, and Vigdis By Kampenes: "*An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry*", Proc. the 10th IEEE International Metrics Symposium (Metrics), Chicago, USA, 2004, 72-83

SP4 Torgeir Dingsøyr and Finn Olav Bjørnson: "*Using Open Space Technology as a Method to Harvest Domain Knowledge*", Proc. 7th International Workshop on Learning Software Organizations (LSO), Kaiserslautern, Germany, 2005, 102-106

SP5 Kari Smolander, Kurt Schneider, Torgeir Dingsøyr, Finn Olav Bjørnson, Pasi Juvonen and Päivi Ovaska: "*Future studies of Learning Software Organizations*", Professional Knowledge Management, Springer LNAI 3782, 2005, 134-144

SP6 Geir K. Hanssen, Hans Westerheim, Finn Olav Bjørnson: ”*Using Rational Unified Process in an SME – A Case Study*, Proc. EuroSPI'05 conference, Budapest, Springer LNCS 3792, 2005, 142-150

SP7 Jingyue Li, Finn Olav Bjørnson, Reidar Conradi, and Vigdis By Kampenes: "An *Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry*", Journal of Empirical Software Engineering, 11(3), 2006, 433-461

Table 2, gives an overview of how the papers relate to our overall research themes. The X indicates to which theme the paper belongs, while (x) indicates that the paper partly belongs to the indicated theme.

Table 2: Papers vs. Research Theme

Paper	RT1	RT2	RT3	Comment
P1		X		Initial results from an attempt to create a knowledge sharing tool.
P2		X		Results from adapting RUP to the software process of a medium sized company.
P3			X	Results from improving a mentor program in a medium sized company.
P4		(x)	X	Results from two companies using the process workshop approach to define their software process.
P5		X		Results from five companies adapting the RUP.
P6			X	Results from a controlled experiment to test an adaptation of the PMA.
P7	X			Results from a systematic review of the literature.
SP1		(x)		Results from a pre-study on COTS based development.
SP2		(x)		Results from an initial explorative study on COTS based development.
SP3		(x)		Main results from the study reported in P2.
SP4			(x)	Suggestion of a method for eliciting domain knowledge, using open space technology.
SP5			(x)	Adaptation of SP4 as part of a larger paper on future directions for learning software organizations.
SP6		(x)		The use of the rational unified process in an SME
SP7		(x)		SP3 adapted for publication in an international journal.

1.6 Contributions

The main contributions of this thesis are listed below.

C1 An overview of the research literature on empirical studies of knowledge management in software engineering.
Through a systematic review we created an overview of the research literature to identify what had been investigated and where the holes in the field were. We believe this work is a good building block to establish a complete and systematic overview of the scientific studies within the field.

C2 A method for tailoring the Rational Unified Process to the development process of a software consulting company.
Through an action research project we gained insight into the process of tailoring the Rational Unified Process to the development process of a medium

sized software company. Our results were contrasted and strengthened by two other case studies and a systematic literature study.

C3 Improvements of the Process Workshops method by contextualization.
Through two action research project where the Process Workshops method was applied to define the software process for two companies, we gained deeper understanding of how the company context affected the results and execution of this method.

C4 Improvement of the root-cause analysis phase of the lightweight Post Mortem Analysis for more effective project retrospectives.
We proposed changes to the post mortem analysis which we tested in a controlled experiment. The result was a more effective method that discovered deeper and more explicit causes for project problems. We also discovered that the revised method was less dependent on professional facilitators.

C5 Proposal of methods to increase the learning effect of mentor programs in software engineering.
Through an action research project in a medium sized software company, we gained deeper insight into how knowledge was shared in a mentor program, and we proposed several modifications to the program that could increase the learning effect.

Table 3 shows the connection between research questions, papers and contributions.

Table 3: Research questions vs. Papers and Contributions

Research Questions	Contributions	Papers	Focus
RQ1.1	C1	P7	KM in SE
RQ1.2	C1	P7	KM in SE
RQ2.1	C2, C3	P1	EPG
RQ2.2	C2	P2, P5	RUP
RQ2.3	C3	P4	PWS
RQ3.1	C5	P3	Mentoring
RQ3.2	C3	P4	PWS
RQ3.3	C4	P6	Retrospectives

1.7 Thesis Structure

The structure of the rest of the thesis is as follows:

Chapter 2: In this chapter we briefly present the field of software engineering and the role of software process improvement. We focus particularly on the use of knowledge management in software engineering, or the learning software organization as it is also known. We also give an overview of research methods in software engineering, and a detailed description of the research methods used in this thesis.

Chapter 3: Here we present the research method we have used for our different studies with arguments for why this is suited for our cases. We explore our research themes and our chosen research questions. We also describe the different contexts of our studies.

Chapter 4: We present the main results of our studies. The chapter first explores all the individual studies. We then sum up our contributions.

Chapter 5: We discuss our findings within our three major research themes. Comparing them with our contributions and the state-of-the-art.

Chapter 6: We sum up the main findings from the discussion, and outline possible further work in the field of knowledge management in software process improvement.

Appendix A: We present the seven papers that have been submitted or published that contain material this thesis is based upon.

Appendix B: We present abstracts of the seven papers that were omitted from the final thesis.

2 *State of the Art*

In order to provide an overview of the context we have been working in, we briefly present some definitions of what software engineering is. We then move closer to our focus area by looking at previous work in software process improvement and previous work in knowledge management. To contextualize further we present some key theories from organizational learning before taking a closer look at how knowledge management has been applied in software engineering. Finally we present an overview of previous research on the methods we have tested during our research, and present overviews on the research methods we have applied with their strengths and weaknesses.

2.1 Software Engineering

Finkelstein and Kramer (2000) describes *software engineering* as the branch of system engineering concerned with the development of large and complex software intensive systems. It is concerned with the processes, methods, and tools for the development of software intensive systems in an economic and timely manner. Fenton and Pfleeger (1997) define that software engineering activities or phases include managing, estimating, planning, modeling, analyzing, specifying, designing, implementing, testing, and maintaining.

Philippe Kruchten (2001) discusses why software engineering differs from structural, mechanical, and electrical engineering due to the soft, but unkind nature of software. He suggests four key differentiating characteristics:

- Absence of fundamental theories or at least practically applicable theories makes is difficult to reason about software without building it.
- Ease of change encourages changes in software, but it is hard to predict the impact.
- Rapid evolution of technology does not allow proper assessment, and makes it difficult to maintain and evolve legacy systems.
- Very low manufacturing costs combined with ease of change have led the software industry into a fairly complex mess.

The term “software engineering” was brought into common use at the 1968 NATO conference on software engineering (Naur, 1969). Even before the conference, back in the days of the punching cards, the development of software was regarded as problematic, and a trend for commercial and governmental systems to be delivered late, over budget and lacking functionality was becoming apparent. The term software engineering was chosen deliberately for the conference in order to be provocative. With the negative trends becoming apparent, the field felt it was necessary for software development to be performed with the rigor and discipline associated with other branches of engineering (Edwards, 2003).

Fast forward 40 years, reading the monthly "Inside Risks" in Communication of the ACM (CACM, 2007), not much seem to have changed. Companies are still reporting that systems are delivered late, over budget and lacking functionality. Edwards (2003) states that if the general expectation within software engineering is that software will not work properly and a crisis-filled environment are reasonable indications, then software engineering is indeed a profession in a continuing state of crisis.

2.2 Software Process Improvement

That is not to say, however, that nothing has been done to improve matters. Many systematic attempts have been made to produce software that is more reliable and of higher quality. Starting in the early 1990's a new set of ideas on how to improve quality and productivity within software engineering was being developed under the notion of Software Process Improvement (SPI). Today, SPI has become one of the dominant approaches to improve quality and productivity in software engineering (Aaen, 2001). SPI is an applied academic field drawing on its roots in both the software engineering and information systems disciplines. The field takes a managerial approach rather than dealing directly with the techniques used to write code, and it deals primarily with managing software firms to improve their practice (Hansen, 2004).

Glass (1999) provides an overview of seven initiatives for improved quality in software engineering in an article in Communications of the ACM: structured techniques, fourth generation programming languages, computer aided software engineering, formal methods, cleanroom methodologies, process models and object-oriented technology. Common for most of these initiatives, or technologies as Glass calls them is that they show promising results, but there is a lack of research, and more studies are needed to properly determine how they work in practice and what the actual benefits are. For the field of SPI, our interest is on what Glass terms the "process models". These are the techniques that have the greatest relevance to the management aspects of software engineering as opposed to the pure technical aspects. We believe that if improvements focus purely on the technical aspects, what will likely be achieved is at best "islands of knowledge", which is a widely recognized problem in knowledge management.

The process models can be used to divide the field into two approaches. The first approach tries to improve the process through standardization, examples here are the Capability Maturity Model (CMM) (Humphrey, 1989; Paulk, 1993; Paulk, 1995), the ISO 9000 standard (Braa, 1994; Hoyle, 2001), and the Software Process Improvement and Capability dEtermination, or SPICE (SPICE, 2007). An alternative to standardization is a more bottom up approach involving the developers in defining their own processes. This approach has its roots in the Total Quality Management (TQM) line of though (Pascale, 1991; Deming, 2000), and is known in software engineering is the Quality Improvement Program (QIP) which was pioneered at the Software Engineering Laboratory at NASA’s Goddard Space Flight Center (Basili, 1992; Basili, 1995)

An attempt at establishing an overview of the SPI field is described in (Aaen, 2001). Aaen et al. describes a survey of the state-of-the-art knowledge on SPI, and position SPI in the landscape of strategies aimed at maturing software organizations. They identify three fundamental concerns in SPI, the principles used to Manage the intervention, the Approach taken to guide the intervention, and the Perspectives used to focus the intervention on the target (MAP for short). Concerning the management of the intervention they identify three key factors: the organization, how it is planned, and the feedback on the effort. Within the approach, key factors are: the evolution of the intervention, the norms followed, and the commitment of employees. The perspective is guided by processes, competence and context. They go on to classify current SPI literature within this MAP framework. One finding is that the literature on SPI seems to focus primarily on aspects related to norms for classification, and compliance to these norms. Areas that have not received adequate attention by the research community include: the organizational context, management commitment, the intervention process, and the building of competence.

Conradi and Fuggetta (2002) posits that software process improvement efforts are characterized by two dichotomies: discipline vs. creative work and procurer risk vs. user satisfaction. They define discipline as the introduction and adherence to more structured work processes, and creativity as emphasizing that software development relies on a collaborative design process known as participatory development. They also state that “software work, like other design work, is not like mechanized or disciplined manufacture. It has a strong creative component involving human and social interaction that cannot be totally pre-planned in a standardized and detailed process model”.

Another overview of research in the field of software process improvement is given by Hansen et al. (2004), they reviewed 322 contributions to the SPI literature in order to establish an overview of research in the field and categorized them according to a simple framework. Whether the papers were prescriptive (suggesting solutions without validation), descriptive (describing an implementation of a method or technology in practice), or reflective (reflecting findings from practice with academic theory). They conclude that the field is heavily biased towards prescriptive contributions, and that the field is dominated by the Capability Maturity Model (CMM) approach. They make a call for more reflective contributions in the field in order to strengthen it.

The finding that field is biased toward prescriptive contributions is mirrored by Glass et al. (2004), who presents an overview of the literature in the whole software engineering field. Their finding is that in software engineering, formulative and descriptive research dominates with only 14% of the studies being evaluative. The research methods most prominently used are found to be conceptual analysis and concept implementations.

2.3 Knowledge Management

Knowledge management is a large interdisciplinary field, encompassing anthropology, social psychology, organization theory, and economics (among others), and it is beyond the scope of this thesis to engage in the ongoing discussion on what knowledge management "is". Instead, we will provide some definitions in use within the field and refer to survey articles in the field with their major findings.

In the first paper in the first number of the first issue in the Journal of Knowledge Management, Wiig (1997) claims the foundation for knowledge management was established and emerged in many organizations in different disguises. His timeline of knowledge management starts in 1975 when Chapparal Steel based its internal organizational structure and corporate strategy to rely directly on explicit management of knowledge. Knowledge management slowly gained momentum and started growing rapidly during the 1990's.

One of the leading authors in the field of knowledge management, Davenport (1998), defines knowledge management as "a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company's knowledge".

Closely related to knowledge management is the term "organizational learning". Organizational learning differs from individual learning in two ways according to Stata (1996). First, it is based on shared insight, knowledge and shared models. Second, it is also based on institutional mechanisms like policies, strategies, explicit models and defined processes in addition to the memory of the participants in the organization. These mechanisms are often referred to as the culture of an organization and are subject to change over time.

A widely cited article concerning strategies for knowledge management is (Hansen, 1999) in which the author refers to two main strategies for managing knowledge:

- *Codification* – to systematize and store information that represents the knowledge of the company, and make this available for the people in the company.
- *Personalization* – to support the flow of information in a company for example by storing information about knowledge sources, like a "yellow pages" of who knows what in a company.

In the introduction to the book *Challenges and Issues in Knowledge Management* (Buono, 2005), in the field of management consulting, Buono and Poulfelt claim that the field is moving from first to second generation knowledge management. In first generation knowledge management, knowledge was considered a possession, something that could be captured, thus knowledge management was largely a technical issue on how to capture and spread the knowledge through tools like management information systems, data repositories and mechanistic support structures. The second generation of knowledge management is characterized by knowing-in-action. Knowledge is though of as a socially embedded phenomenon, and solutions have to consider complex human systems, communities of practice, knowledge zones, and organic support structures. The change in knowledge management initiatives is seen to go from a planned change approach to a more guided changing approach.

Coming from the field of management consulting, Christensen (2005) performed a literature review focusing on special journal issues on knowledge management from 1995-2003. He performed a content analysis of 50 identified papers focusing on knowledge management context, knowledge management outcomes, empirical setting and the key drivers for knowledge management. The finding was that KM writings seem to focus on how to create knowledge and to a lesser degree, how to transfer knowledge. The categories that did not receive adequate coverage were integration, production, measurement, retention and reflection. A second finding was that the drivers for both knowledge creation and knowledge transfer were generic and to a large degree overlapping. He goes on to explore knowledge management in practice through 10 managers from industry and compares his results to the results of the theoretic study. The main conclusion is that KM theory does reflect, in generic terms, the practices that support KM activities, but the challenge is to observe this practical application of generic drivers, which often is difficult to observe in practice.

Another overview on knowledge management, coming from the field of information systems, is given by Alavi and Leidner (2001). One of the major challenges in KM according to them is to facilitate the flow of knowledge between individuals so that the maximum amount of transfer occurs. They also conclude that no single or optimal solution to organizational knowledge management can be developed. Instead a variety of approaches and systems needs to be employed to deal with the diversity of knowledge types. Knowledge management is not a monolithic but a dynamic and continuous phenomenon.

Earl (2001) has made a framework for classifying work in knowledge management (see Table 4). He defines the different approaches as schools of knowledge management. The schools are broadly categorized as “technocratic”, “economic” and “behavioral”. The technocratic school consists of three schools: The systems school, which focuses on technology for knowledge sharing, the cartographic school, which focuses on tools to enable people to locate people with the right knowledge, and finally the engineering school, which focuses on processes and knowledge flows in organizations.

The economic school focuses on how knowledge assets relates to income in organizations.

The behavioral school consists of three sub-schools: The organizational school focuses of networks for sharing knowledge, the spatial school focuses on how office-space can be designed to promote knowledge-sharing and finally the strategic school focuses on how knowledge can be seen as the essence of a company’s strategy.

Table 4: Earl's schools of knowledge management.

	Technocratic			Economic	Behavioral		
	Systems	Cartographic	Engineering	Commercial	Organizational	Spatial	Strategic
Focus	Technology	Maps	Processes	Income	Networks	Space	Mindset
Aim	Knowledge bases	Knowledge directories	Knowledge flows	Knowledge assets	Knowledge pooling	Knowledge exchange	Knowledge capabilities
Unit	Domain	Enterprise	Activity	Know-how	Communities	Place	Business

2.4 Theories of Organizational Learning

In cognitive and organization science, we find many models on how knowledge is transferred or learned at an individual and organizational level. We present three theories that are widely referred to: Nonaka and Takeuchi´s theory of knowledge creation, Wenger's theory of communities of practice and the double-loop learning theory of Argyris and Schön.

Nonaka and Takeuchi's (1995) theory on knowledge creation is based on the distinction between explicit and tacit knowledge.

- *Explicit knowledge* is knowledge that is transmittable in formal, systematic languages. It can be articulated in formal languages, including grammatical statements, mathematical expressions, specifications, manuals and so forth. It can be transmitted across individuals formally and easily.

- *Tacit knowledge* is personal and context-specific, and is therefore difficult to formalize and communicate. It is personal knowledge that is embedded in individual experience and involves intangible factors such as personal belief, perspective, and value system. Tacit knowledge is difficult to communicate and share in the organization and must thus be converted into words or forms of explicit knowledge.

The very idea in software engineering is to explicate knowledge in the forms of programs to be executed on computers. Software developers spend great effort developing programs, specifications, and models, while at the same time participating in close people-to-people interactions as members of software teams.

According to Nonaka and Takeuchi organizational knowledge is created during the time the "conversion" between these forms takes place, i.e. from tacit to explicit and back again into tacit. Knowledge conversion is a "social" process between individuals and is not confined to one individual. Assuming that knowledge is created through interaction between tacit and explicit knowledge, four different modes of knowledge conversion are possible, see Figure 2.

1. From tacit knowledge to tacit knowledge: Socialization
2. From tacit knowledge to explicit knowledge: Externalization
3. From explicit knowledge to explicit knowledge: Combination
4. From explicit knowledge to tacit knowledge: Internalization.

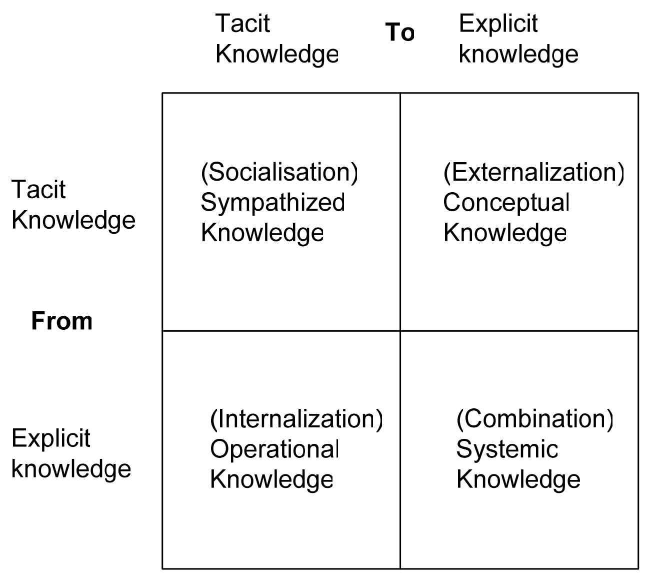
Tacit Knowledge
To
Explicit knowledge
Tacit Knowledge
From
Explicit knowledge
(Socialisation) Sympathized Knowledge
(Externalization) Conceptual Knowledge
(Internalization) Operational Knowledge
(Combination) Systemic Knowledge

Figure 2: Four Modes of Knowledge Conversion

According to Nonaka and Takeuchi knowledge passes through different modes of conversion, which makes the knowledge more refined, and also spreads it across different layers in an organization.

Another theory was proposed by Wenger (1998). “Communities of practice” (CoP) are defined as “Groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis” (Wenger, 2002). Knowledge exists within these communities in the way the participants work and act, and is expanded and shared through what Wenger calls participation and reification. Participation refers to the participation in the community, interaction and active involvement. Reification refers to the process of creating artifacts from the community. The communities are often different from normal business units in that they are informal and self managed.

Knowledge sharing between communities is referred to as boundary relations. The theory specifies two types of boundary relations: Contact through participation is called brokering and contact through reification is called boundary objects.

According to Wenger (1998), a practice can be described as “shared histories of learning”. Wenger makes three points in this regard. 1) Practice is not stable, but combines continuity and discontinuity. 2) Learning in practice involves three dimensions; practices are histories of mutual engagement, negotiation of an enterprise, and development of a shared repertoire. 3) Practice is not an object but rather an emergent structure that persists by being both perturbable and resilient.

The CoP theory separates learning into different levels, the individual, community and organizational level. For individuals the learning takes place in engaging in and contributing to a community. For communities, learning is defined as refining the practice. On the organizational level, learning is to sustain interconnected communities of practice.

In their theory on learning, Argyris and Schön (1996) distinguish between what they call single and double-loop learning in organizations. In single-loop learning if consequences of actions aren't met, you change your actions slightly to achieve the desired results. It is a feedback-loop from observed effects to making some changes or refinements that influence the effects, see Figure 3.

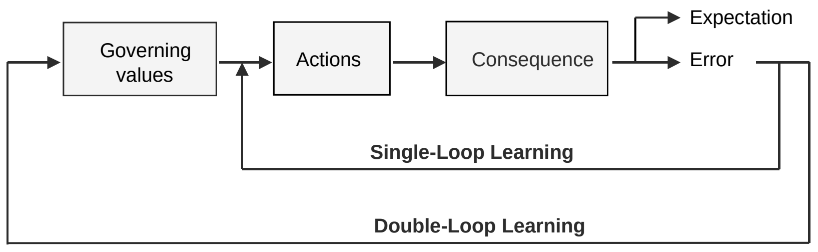

Figure 3: Single and double loop learning

Double loop learning, on the other hand, is when you take the time to understand the factors that influence the effects, and the nature of this influence, which is called the "governing values" (Argyris, 1990). This could be to understand why a process is usable, that is: Which premises must be satisfied for it to be worthwhile. To make changes based on this type of understanding will be more thorough.

2.5 Knowledge Management in Software Engineering

A software engineering company who actively uses knowledge management is often referred to in the literature as a "learning software organizations". An organization that have to "create a culture that promotes continuous learning and fosters the exchange of experience" according to Feldmann and Althoff (2001). Another definition, by Dybå (2001) puts more emphasis on action: "A software organization that promoted improved actions through better knowledge and understanding". Edwards (2003) claims that knowledge management in software engineering is somewhat distanced from mainstream knowledge management, and claims the reason for this lack of "visibility" of software engineering in the wider knowledge management literature is a tendency for discussion of such topics to take place only at conferences for the software engineering community.

In a systematic review we carried out as part of this thesis, we found that from 1999 and onwards, there has been an increase in publications on experience from knowledge management efforts in software engineering. 1999 was also the year the first workshop on "learning software organizations" was organized in conjunction with the SEKE conference. This workshop has been one of the main arenas for empirical studies as well as technological development related to knowledge management in software engineering. Other arenas for knowledge management in software engineering includes a special issue of IEEE Software (Lindvall, 2002), and the book “Managing Software Engineering Knowledge” (Aybüke, 2003).

There have been some previous attempts at establishing an overview of works published on knowledge management in software engineering. Rus et al. (2001) present an overview of knowledge management in software engineering, focusing on motivations for knowledge management, approaches to knowledge management and factors that are important when implementing knowledge management strategies in software companies. Lindvall et al. (2001) describe types of software tools that are relevant for knowledge management, ranging from document and content management tools to collaboration tools and tools for competence management. Dingsøyr and Conradi (2002) surveyed the literature for studies of knowledge management initiatives in software engineering, and found eight lessons learned reports, which are characterized after what actions companies took, what the effects of the actions were, what benefits are reported and what kind of knowledge management strategies were used.

The subject of previous studies of knowledge management in software engineering is the focus of our first research theme, and is reported in greater detail throughout the thesis. We will just briefly state our main conclusions from our systematic review here in order to provide a brief overview of the field. Using the framework of Earl (2001) described in section 2.3, we conclude that the studies on knowledge management in software engineering is mainly related to the technocratic and behavioral schools with a large bias towards the technocratic side. Schools with particularly poor coverage include the economic, spatial and cartographic schools. Within the schools that were covered there was little overlap between the different studies. We also found that the majority of papers were reports of lessons learned and not qualified as scientific studies. From the papers that could be classified as scientific, more than half were case studies. For our complete review see the enclosed paper P7 in appendix A.

2.6 Selected methods and technologies

During the work of this thesis some selections were made with regard to possible methods and technologies that could be studied in our companies. These selections were made in cooperation between the case companies and the researchers. We strived for a good compromise that would allow the company to invest in methods they saw as beneficial, and the researchers to study methods relevant to our research goals. The complete rationale for each company and choice of method is presented in chapter 3. In order to follow our discussions and research questions relating to these methods, we will now present a brief introduction to the methods we have studied during the work of this thesis. With regards to the codification strategy we studied the rational unified process and the process workshops method. Concerning personalization approaches we studied knowledge created during process workshops, a retrospective method called the post mortem analysis, and a mentor program.

2.6.1 Rational Unified Process

The Unified Process (Jacobson, 1999) and the commercial variant, the Rational Unified Process, RUP (Krutchen, 2000) are comprehensive process frameworks for software development projects. RUP defines a software development project as a set of disciplines, e.g. requirements handling, implementation etc., running from start to end trough a set of project phases. A project is performed by a group of actors, each having one or more well defined roles. Each role participates in one or more activities producing one or more artifacts. A discipline can run in iterations, that is, repetitions within a phase. Activities, roles and artifacts are the basic process elements of RUP. The concept of role, activity and artifact are central in RUP. A *role* performs an *activity* to produce or update an *artifact*.

However, RUP is a comprehensive framework, meaning that it is a more or less complete set of process elements that has to be tailored to each case as no project needs the complete set of elements. Jacobson, Booch and Rumbaugh (1999) says on p.416: *"It [RUP] is a framework. It has to be tailored to a number of variables: the size of the system in work, the domain in which that system is to function, the complexity of the system and the experience, skill or process level of the project organization and its people."* Further on they say: *"Actually, to apply it, you need considerable further information."* So, it is clear that RUP needs to be tailored, downscaled and specialized to the context of use.

There exists a set of guidelines for tailoring and adoption of RUP; one book that specifically targets the issue (Bergström, 2003) and one book that covers the issue to some detail (Kroll, 2003). Additionally there exists a guideline documented through a website. In addition there are some guidance in the RUP documentation itself or RUP-related books, however these guidelines tends to be superficial. Despite the existence of these guidelines we have not been able to find any experience reports evaluating their outcome and suitability.

The process of adapting RUP can possibly take many forms. IBM Rational, the provider of RUP has defined the Process Engineering Process (PEP). This is a comprehensive adaptation process requiring a fairly big amount of resources (people and time). This may very well be appropriate for larger companies, but for the small ones this process may be too expensive. Adaptation of a framework, such as RUP, can take one of (at least) three approaches. The first is to do it in one step, for each project, thus representing a heavy job in each case. This can be justified for large projects. This approach may be called situational method engineering, as defined by ter Hoefstede and Verhoef (1997). The second approach is to do an up-front adaptation producing a subset of the framework, still being a framework, but now tuned to the organizations general characteristics (technology, customers, domain, traditions etc.). This is the intentional process of PEP and may be called method engineering, as defined by Brinkkemper (1996). The thirds approach is to first identify and describe a set of recurring project types. Having knowledge of characteristics and differences of these types, an adaptation is done for each type. No matter which approach being used; in the last step, a final adaptation is done to each case or project.

2.6.2 Process Workshops

When companies choose to design their own development processes, one option is to assign the task to a group of expert “process engineers” as described by Becker-Kornstaedt (2001). One or more process engineers elicit process data from interviews, documents, surveys, e-mails and observation, and then interpret this data to produce a process model. This approach relies heavily on the experience and skill of the process engineer. Therefore, without any structured method, quality and repeatability cannot be ensured. It is, however, unlikely that the use of qualitative methods alone can compensate for experience in process modeling and software engineering according to Carvalho (2005).

An alternative to using process engineers is to involve the employees more in designing the process models, for example through workshops (Ahonen, 2002; Moe, 2005). This type of work takes up the heritage from employee participation in organizational development, a part of “Scandinavian work” tradition as well as in most work on improvement, from the Total Quality Management principles (Deming, 2000) to the knowledge management tradition in Communities of Practice (Wenger, 1998).

In one of the studies reported in this thesis, we used a method called process workshop (Dingsoyr, 2005), which is a method to define current or future processes in a process guide. The method is designed to involve the users of the future process in discussing and defining the processes. It ensures that people discuss how they work – which fosters learning even before the process guide is available in the company. It also assures quality – the process guide is developed by people who know how to do the work; it does not describe how external consultants or senior staff imagine what “ideal” development processes should look like.

The process workshop method was designed as a lightweight method to help facilitate the development of process guides. Apart from the original introduction of the process workshop (Dingsoyr, 2005) and a Finnish application of the same method (Pikkarainen, 2005), there is little empirical evidence on the practical application of this method.

2.6.3 Project Retrospectives

According to Rising et al. (2003), retrospective analysis as a method for learning from work experience was identified in 1988 by Joseph Juran and named ”Santayana review” in homage to the philosopher George Santayana. Since then, many organizations have used many variations of the method and under many different names. Dingsøyr (2005) lists the most common names for retrospective analysis: ”project retrospectives”, ”post mortem analysis”, ”postproject review”, ”project analysis review”, ”quality improvement review”, ”autopsy review”, ”after action review”, and ”touch down meetings”.

Dingsøyr (2005) discusses the importance of retrospective analysis as a method for sharing knowledge in software projects and gives an overview of the methods of retrospective analysis that are employed in the field of software engineering. In particular, Dingsøyr presents three lightweight methods of retrospective analysis, which are presented by Whitten (1995), Collison and Parcell (2001), and Birk et al. (2002). He compares the three methods with respect to: particpiants, the need for homework, the type of discussion and the output of the analysis.

Another one comparing the outcomes of retrospectives is Desouza *et al.* (2005) they compare two kinds of output from retrospective analysis: traditional reports and stories. They also identified four factors that should affect the choice of writing the result of the PMA as a report or as a story: (1) the nature of the project, (2) the cost you are willing to bear, (3) how much organizational impact is desired, and (4) what lessons you wish to convey.

Myllyaho *et al.* (2004) conducted an extensive literature review within the software engineering and management literature, with the aim of reviewing retrospective analysis as a project-based learning technique. The results suggest that the use of retrospective analysis is well worth the effort, and that a simplified or ’lightweight’ version of PMA can be beneficial when time is a factor.

In one of our major contributions to this thesis we take our starting point in the method suggested by Birk et al. (2002). The aim of this method is to bring together project participants and have them discuss what went well and what could be improved, and to analyze the root causes. The method uses two techniques to carry out the PMA. To discover the positive and negative experiences, they use a focused brainstorm method called the KJ-method (Scupin, 1997), resulting in affinity diagrams. To analyze the causes of these experiences, they perform root cause analysis using fishbone diagrams (also known as Ishikawa diagrams, in reference to their inventor Dr. Kaoru Ishikawa, a Japanese quality control statistician).

2.6.4 Mentoring

In one of our studies we investigated mentoring, we did not find any background material in the software engineering field, so we based our research on management theory.

Kram (1985) suggests that existing theory predicts that effective mentoring should be associated with positive career and job attitudes. In a literature review, Ragins et al. (2000) show that empirical studies support this proposition. They also present results from a survey that indicate that persons in dissatisfying or marginally satisfying mentor relationship express the same or worse attitudes than people not involved in a mentor relationship at all. One of their conclusions is that it is clear that good mentoring may lead to positive outcomes, but bad mentoring may be destructive and in some cases worse than no mentoring at all.

What is a mentor and protégé? According to Kram (1985), mentors are generally defined as “individuals with advanced experience and knowledge who are committed to providing upwards mobility and career support to their protégé”. A protégé literally means “a person under the patronage, protection, or care of someone interested in his career or welfare” (Webster's, 1989). This is usually a younger employee who lacks experience in one or more fields.

According to a literature review of mentoring by Ragins et al. (2000), comparisons of non-mentored and mentored individuals yield the consistent result that individuals with informal mentors report greater career satisfaction, career commitment and career mobility than individuals without mentors. Many organizations have attempted to replicate the benefits of informal mentoring by developing formal mentor programs. Yet formal and informal mentoring relationships vary on a number of dimensions:

- Informal mentor relationships often arise through a mutual developmental need, and often spring from mutual identification. The mentor may view the protégé as a younger version of themselves and the protégé may view the mentor as a role model. This mutual identification contributes to a closeness and intimacy of the mentor program which is often cited in mentoring literature (Kram, 1985). An informal mentor program is often unstructured and the participants meet as often and as long as is desired. Such an informal mentor relationship usually lasts between three and six years. The purpose of informal mentoring relationships is often the achievement of long term career goals for the protégé.

- In contrast, formal mentoring relationships usually spring from a third party assigning the mentor and protégé to the relationship. This may lead to people entering into these relationships not because of mutual need but to meet organizational standards. Meetings in a formal mentoring relationship are often sporadic or specified in a contract at the start of the program, and their duration is often from six months to one year, much shorter than informal relationships. Because of this short time span, the purpose of formal mentoring is often the achievement of short term career goals.

Kram and Hall (1989) claim that mentor activities are “prime and untapped resources in creating the learning organization”. Allen and Eby (2003) claim that mentors as well as protégés should benefit from a mentoring program including learning about “new technologies” and receiving updates on issues at other levels of the organization. But they also report that there is still a need to empirically examine these issues.

2.7 Research Methods in Software Engineering

Traditionally, software engineering has focused on coming up with new tools and techniques without much validation beyond concept analysis and concept implementation. In a review of the field, Glass et al. (2004) found that only 13.8% of the literature on software engineering could be classified as evaluative, the remaining papers being mostly formulative or descriptive. The main research methods employed within the field was classified as conceptual analysis and concept implementation, representing a total of 71.2% of the papers.

In other words, there are a lot of papers and books with huge amounts of good advice, among which no one knows which ones are truly good, and which ones are merely rituals serving no purpose. In order to remedy this situation, the field of empirical software engineering has emerged.

Empirical research is based on the scientific paradigm of observation, reflection and experimentation as a vehicle for the advancement of knowledge (Endres, 2003). Empirical studies may have different purposes, being *exploratory* (investigating parameters or doing a pre-study to decide whether all parameters of a study are foreseen), *descriptive* (finding distribution of a certain characteristics), or *explanatory* (investigating why certain phenomena happen).

There are three types of research paradigms that have different approaches to empirical studies (Creswell, 1994; Seaman, 1999; Wohlin, 2000; Creswell, 2003):

- *Qualitative research* is concerned with studying objects in their natural setting. A qualitative researcher attempts to interpret a phenomenon based on explanations that people bring to them (Denzin, 1994).
- *Quantitative research* is concerned with discovering causes noticed by the subject in the study, and understanding their view of the problem at hand. A quantitative study is mainly concerned with quantifying a relationship or to compare two or more groups (Creswell, 1994). The quantitative research is often conducted through setting up controlled experiments or collecting data through case studies or surveys.
- The *mixed-method approach* is evolved to compensate for limitations and biases of the above strategies, seeking convergence across other methods. The combination of quantitative and qualitative methods is usually more fruitful than either in isolation (Seaman, 1999). How to combine the qualitative and quantitative method in the design is described by (Basili, 1986) and further discussed in (Seaman, 1999).

Depending on the purpose of the evaluation, whether it is techniques, methods, or tools, and depending on the conditions for the empirical investigation, the empirical research strategies can be classified into different categories. Zelkowitz and Wallace (1998) summarized 12 technology validation models and grouped these models into categories according to the data collection methods: observational, historical, and controlled. The validation models in (Zelkowitz, 1998) include models to examine both the projects (e.g., case study, project monitoring, and field study) and products (e.g., static analysis, simulation, and dynamic analysis).

According to the framework of Zelkowitz and Wallace, we have used research methods from all three major categories: observational, historical, and controlled. In the observational category, we followed several case companies. However, since we were deeply involved with the companies and their choice of methods for improvement, we chose not to classify these projects as case studies, but rather as individual action research studies. In the historical category, we performed a literature study, but with added strictness to the method to make it a systematic review. In the controlled category we performed a controlled experiment. We now describe the general characteristics of these research methods. See chapter 3 for a detailed description of how they were applied in our studies.

2.7.1 Action Research

According to Baskerville (1999) action research is an established research method in use in the social and medical sciences since the mid-twentieth century. Towards the end of the 1990s it began growing in popularity for use in scholarly investigations of information systems.

Avison et al. (1999) states that action research is unique in the way it associates research and practice. Research informs practice and practice informs research synergistically. Action research combines theory and practice (and researchers and practitioners) through change and reflection in an immediate problematic situation within a mutually acceptable ethical framework.

Action research presumes that complex social systems cannot be reduced for meaningful study, nor can sociological experiments ever achieve repeatability. According to Baskerville (1999): "The fundamental contention of action research is that a complex social process can be studied best by introducing changes into that process and observing the effects of these changes". This definition fits closely with the goal of SPIKE to do "empirical studies where methods and techniques are tried out in company projects". By introducing changes we fulfill the need of the companies to improve their processes and according to Baskervilles definition, observing the effect of our changes enables us to study and understand the complex process that is software development.

Baskerville goes on to say that a clear area of importance in the ideal domain of action research is new or changed systems development methodologies. He states that action research is one of the few valid research approaches that we can legitimately employ to study the effects of specific alterations in system development methodologies in human organizations.

The most prevalent action research description (Susman, 1978) details a five phased, cyclical process. The approach first requires the establishment of a cooperation between practitioners and researchers, called a client-system infrastructure or research environment. Then, five identifiable phases are iterated: diagnosing, action planning, action taking, evaluating and specifying learning. Figure 4 illustrates this action research structural cycle.

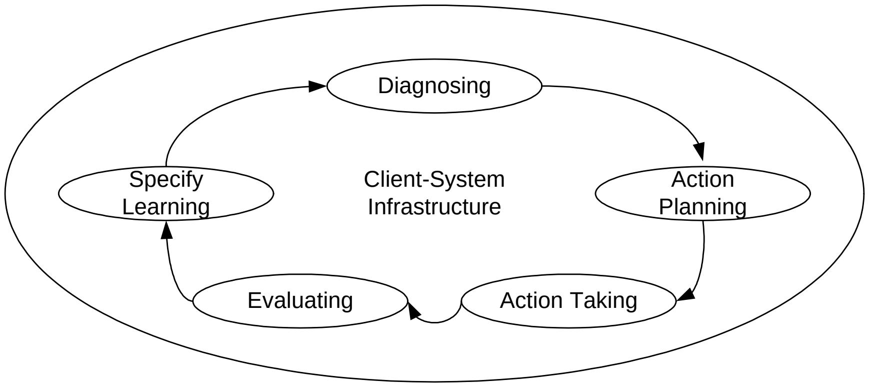

Figure 4: Action Research

In order to achieve valid action research Davison et al. (2004) suggests following five principles (Table 5). These principles have formed the basis for our action research studies. In order to differentiate the different studies, we also classified them according to the three aspects of control structures suggested by Avison et al. (2001) (Table 6).

Table 5: The five principles of canonical action research, by Davison et al.

Principles of canonical action research
1. The principle of the researcher-client agreement.
2. The principle of cyclical process model.
3. The principle of theory.
4. The principle of change through action.
5. The principle of learning through reflection.

Table 6: Forms and Characteristics of the major AR control structures, by Avison et al.

Control aspect	Forms	Characteristics
Initiation	Researcher	Field experiment
	Practitioner	Classic action research genesis
	Collaborative	Evolves from existing interaction
Authority	Practitioner	Consultative action warrant
	Staged	Migration of power
	Identity	Practitioner and researcher are the same person
Formalization	Formal	Specific written contract or letter of agreement
	Informal	Broad, perhaps verbal, agreements
	Evolved	Informal or formal projects shift into the opposite form

Strengths and Weaknesses of Action Research

Action research has long been advocated as a research method within the information systems field see for example (Baskerville, 1996; Levin, 1998; Mathiassen, 2002), as such the software engineering field can benefit from adopting the method in our research. The major strength of action research is the possibility to work closely with industry and study a phenomenon in depth in a realistic setting.

Weaknesses are related to lack of control of the environment which can lead to problems with generalizing the result. In addition the researchers need to find a balance their roles as consultants and scientists which can create ethical dilemmas. Practical challenges can arise from companies not wanting the scientists to access all projects, leading to selection bias. Industry may want immediate applicable results which may run counter to the goals and practice of research.

2.7.2 Experiments

An experiment is a formal, rigorous and controlled investigation, where the key factors are identified and manipulated (Basili, 1996; Wohlin, 2000). In an experiment, the objective is usually to distinguish between two situations, for example a control situation (before change) and the situation under investigation (after change).

The experiment is usually classified as a quantitative research strategy. This means that the investigation has to measure some factors in a way that produces data that can be reasoned with statistically. This is important because the main tool for distinguishing the effect of a factor comes through hypothesis testing, which is supported by statistical methods. The more valid the data is, the more we can trust the result of the experiment. The trustworthiness of the experiment also depends on the quality of the experiment design. More uncertainties in the design lead to less reliable experiment results.

Experiments are normally done in a laboratory environment, which provides for a high level of control. Subjects are randomly assigned to different treatments, and the objective is to manipulate a small number of variables while all other variables are fixed (controlled). The effect of the manipulation is measured, and based on this a statistical analysis can be performed. In some cases, the investigation environment may prove to be too complex for the investigation to be called a true experiment, which leads to the experiment being known as a quasi-experiment.

Experiments are appropriate to investigate several different questions, for instance to confirm theories, confirm conventional wisdom, explore relationships, evaluate the accuracy of models or validate measures.

There are some well-known techniques used when designing experiments. The reason for employing these is to enhance validity of the investigation and its analysis. The general design principles are randomization, blocking and balancing, and most experiments use some combination of these. Randomization means selecting experiment subjects, objects and test ordering in a manner which does not bias the experiment in a manner that makes generalization of the results wrong. Blocking is used to systematically eliminate factors that probably have an effect on the investigation, but that we are not interested in. An example of doing this is to put subjects that have different characteristics into different blocks according to this characteristic, if this characteristic is not under study but could influence the result. Balancing is just to assign treatments so that each treatment has practically equal number of subjects. This strengthens the statistical analysis of the data.

Experiments can be controlled experiments, or quasi-experiments. Controlled experiments are usually performed as a simulation of a real life situation (in vitro), with a small object of study and a mix of novice and experts involved as subjects. Quasi-experiments are often run in normal working conditions (in vivo), in a large ongoing project with only experts as subjects.

Experiments can also be characterized by the number of teams replicating each project under study and the number of different projects under study. If an experiment is conducted by having one project carried out by one team, it is called a single project. If the project is carried out by more than one team, it is called a replicated project. If there are more than one projects being carried out in the experiment, the single team experiment is called a multi-project variation, while an experiment with several projects and several teams is called a blocked-subject project.

Strengths and Weaknesses of Experiments

Several studies in software engineering have used experiment as a research method. Systematic reviews of the field include (Dybå, 2006; Hannay, 2007). The major strength of a (controlled) experiment is the ability the researchers have to keep the influencing and confounding factors under control. Also of importance is the ability to design the experiment so the result is statistically valid.

The major drawback of this method is the artificial setting which can make it harder to generalize the results. One of the major challenges of this approach is thus to design the experiment so that the setting is realistic, and also recruiting a large enough number of realistic subjects, Sjøberg et al. describe these challenges and their experience with controlled experiments in (Sjøberg, 2003). Even if we get enough realistic subjects and set up a realistic development environment, the experimental setting may contribute to unrealistic experiment conditions and thus influence the results. In other words, great care is needed in designing the experiment in order to get results that are generalizable to a context outside the experiment setting.

2.7.3 Systematic Review

A systematic review differs from a regular literature review in that it puts increased weight on thoroughness and replicability. There are demands placed on research questions, identification of research, selection process, appraisal, synthesis and inferences. Some key features that differentiates a systematic review from a conventional literature review are presented by Kitchenham (2004):

- Systematic reviews start by defining a review protocol that specifies the research question being addressed and the methods that will be used to perform the review.
- Systematic reviews are based on a defined search strategy that aims to detect as much of the relevant literature as possible.
- Systematic reviews document their search strategy so that readers can access its rigor and completeness.
- Systematic reviews require explicit inclusion and exclusion criteria to assess each potential primary study.
- Systematic reviews specify the information to be obtained from each primary study including quality criteria by which to evaluate each primary study.

A systematic review can be divided into three main phases: planning, conducting and reporting the review.

1. Planning:
   a. Identification of the need for a review
   b. Development of a review protocol
2. Conducting the review:
   a. Identification of research
   b. Selection of primary studies
   c. Study quality assessment
   d. Data extraction & monitoring
   e. Data synthesis
3. Reporting the review

The interest in systematic reviews in software engineering has increased lately, resulting in some experience reports from applying the method. Brereton et al. (2007) confirms that the basic steps in systematic review process appears to be relevant in conduction such reviews in software engineering. They do however note that empirical studies in software engineering has a lot of shortcomings, and search facilities lack conformity which serves as a hindrance to systematic literature reviews in the discipline. Dybå et al. (2007) reports similar findings that the general guidelines seems to work well within the software engineering field. They identify the key challenge to be the inclusion of evidence from a variety of perspectives and research methods. There are ample guidelines for including research based on quantitative methods, but there is a lack of advice on how to include results of qualitative studies and studies using mixed-method approaches.

Strength and Weaknesses of Systematic Reviews

The major advantage of systematic reviews is that they potentially can provide information about effects of phenomenon across a wide range of settings and empirical methods. Another advantage is that in the case of quantitative studies, it is possible to combine data through meta-analytic techniques. In addition, since a systematic review requires a predefined search strategy and documentation of this, the result is replicable and it is possible for other researchers to assess the completeness of the search.

The major drawback is that a systematic review requires considerable more effort than a traditional literature review. In addition studies in software engineering are rarely presented in a uniform manner, and thus the combination of results from several studies can be difficult.

2.8 Validity Threats

A fundamental discussion concerning results of a study is how valid they are. Empirical research usually uses definitions of validity threats that originate from statistics and not all the threats are relevant for all types of studies. Wohlin et al. (2000) define four categories of validity threats:

- **Internal Validity:** Is there a causal relationship between the treatment observed and the outcome, or is the connection caused by factors not measured or otherwise considered?
- **External Validity:** Can the results of the study be generalized outside the scope of the study? For instance, when evaluating a tool used in a project, the external validity will consider whether the conclusions are valid in different projects or in the world in general.
- **Construct Validity:** This considers the design of the study. If we have a theory about a new technique being helpful, and we run an experiment that shows the technique to be beneficial, the construct validity concerns whether the experiment is properly designed to say anything about the theory.
- **Conclusion Validity:** This concerns the relationship between treatment and the outcome. For instance, is the effect observed statistically significant?

Different threats have different priorities based on type of research. For example, in theory testing, internal validity is most important, while generalization is not usually an issue. For a case study, Yin (2003) identifies three tactics to improve validity:

- Use multiple sources in data collection and have key informants to review the report in composition to improve construct validity.
- Perform pattern matching (comparing an empirically based pattern with a predicted one especially for explanatory studies) and address rival explanations in data analysis to improve internal validity.
- Use theory in research design in single case studies to improve external validity.

3 Research Approach

We will now specify our research goals from the problem outline given in the introduction. From these we move on to discuss the research method we applied, and details on how the research was carried out.

3.1 Research Goal

Our main focus for this thesis as stated in the introduction was:

How can Knowledge Management be applied to Software Engineering in order to foster Software Process Improvement?

This has been the guiding theme for the research presented in this thesis. Given our setting where company strategies frequently changed, it proved invaluable to have the overall focus of looking at software process improvement efforts from the perspective of knowledge management. Having an overall theme allowed us to adapt our research to the company preferences without deviating too far from our original directions. Given the results from our five main studies, we have broken the overall goal down into three overall themes which allow us to classify our findings.

From our overall research goal, we got the viewpoint for our research in the companies. In cooperation with the participating companies, we agreed on concrete methods and settings. The research questions for each study were closely related to the methodology we investigated and how it could be improved or applied in the given setting. In order to rise above the concrete research questions and come closer to understand the theories of knowledge management, we formulated the overall themes that we will use to bridge the gap between the overall research goal and the explicit research questions.

RT1: Previous research on knowledge management in software engineering

This theme emerged from Study 5, the systematic review of the field. It is an important theme, since in order to advance the field, we need to establish what has already been covered. Our aim with the study was to establish an overview of the field we chose to study, and what methods had been applied to investigate it. Since the focus of our research was empirical studies, we chose to limit this theme to studies with results from actual use in industry. As outlined in chapter 2.5, some attempts had previously been done to establish an overview of the field, but these did not cover the field in a systematic way, and none of them reported results further than 2002. This theme also served to highlight "holes" in the field, and it offered a framework in which to place our own studies. Two concrete research questions were formulated within this theme.

RQ1.1: What concepts have been investigated empirically within the field of knowledge management in software engineering?
In order to identify possible holes in the field, we decided to investigate what had already been covered in the literature and attempt to classify it.

RQ1.2: What are the research methods used in studying knowledge management in software engineering?
In addition to finding *what* had been covered in the research literature, we also wanted to find out *how* it had been covered, since the choice of research method could have influenced the results reported from the field. Thus we introduced this research question to cover the methods used.

For our next two research themes, we chose to use the theories in (Hansen, 1999). They suggest that there are basically two strategies for managing knowledge, codification or personalization. With regards to Nonaka and Takeuchi this concerns explicit and tacit knowledge, and the transitions of codification and socialization. In terms of Wenger's theories we are looking at reification and participation.

RT2: Application of knowledge management to improve the software process through codification of knowledge.

This theme concerns the codification strategy. In order to investigate this strategy, we chose two companies that wanted to improve their software process through different codification initiatives (Study 2 and 3). Our research questions within this theme relates to the methods chosen by the two companies. As researchers we had some influence in what methods they used to define their process, but in the end it was the companies' decision on what they wanted to spend their time and resources on. The following concrete research questions were answered in the studies which relates to codification.

RQ2.1: What do developers want from a knowledge sharing tool?
This research question was formulated early in our work with the industrial partners. In order to improve codification it was interesting to know what artifacts the developers themselves found useful. This information was considered useful irrespective of which method was used to codify it.

RQ2.2: What are the limitations, benefits, prerequisites and cost of tailoring and introducing the Rational Unified Process?
One way to improve the software process is to codify it in a process model. A practical framework for codifying such a process that has gained widespread use in industry is the Rational Unified Process. Despite its popularity there was not much published material on the challenges of adapting such a comprehensive framework to a small or medium sized setting.

RQ2.3: How do available information, company context and goals affect the result of process workshops?
Taking our starting point in a method for eliciting software processes from employees through interaction in workshops, we wanted to expand the contextuality of the method to find out how different contexts affected the outcome.

RT3: Application of knowledge management to improve the software process through personalization of knowledge.

Our third research theme relates to Hansen's (1999) second strategy, namely personalization. We do not take the traditional approach in software engineering and study tools that support the flow of information by connecting knowledge sources, instead we study the learning effects and knowledge transfer that takes place once knowledge sources (people), have been connected. We had more difficulties finding companies that considered using the personalization strategy for their knowledge sharing. We only found one company in our available set of companies that was interested in following this strategy fully. To further our studies in this theme, we decided to observe the tacit knowledge sharing that took place during the workshops of a codification initiative in another company we were involved with. We also planned and executed a controlled experiment to investigate an improvement to a method for project retrospectives. Three concrete research questions have been answered within this theme, the first two were to a large degree chosen based on what the companies we were involved with decided on applying in their setting, the third was chosen independently of companies and tested in an experiment:

RQ3.1: How can knowledge transfer through a mentor program be improved?
One way to transfer knowledge from person to person is the mentoring approach. We wanted to know how it functioned in the context of a medium sized software company, and if we could improve it using theories from the research field.

RQ3.2: How do available information, company context and goals affect the learning effects during execution of process workshops?
Taking our starting point in a method for eliciting software processes from employees, we wanted to expand the contextuality of the method to find out how different contexts influenced the knowledge sharing during the process workshops.

RQ3.3: How can sharing of project experience through project retrospectives be improved?
A way of transferring experience from person to person is project retrospectives. We proposed changes to the brainstorming in the root cause analysis phase of one such method and wanted to test if this was an improvement on the method and if so, what that improvement consisted of.

3.2 Research Process

The research process for this thesis has been iterative and a lot of projects have happened in parallel, mutually influencing each other. The work can roughly be divided into three main directions:

- Three industrial case studies (all using Action Research)
    - Study1: A study of mentoring for transfer of knowledge
    - Study2: A study on codifying the software process through an adaptation of RUP
    - Study3: A study on reaching an agreement on and codifying the software process through the process workshop
- Study4: A study on using and improving the post mortem analysis to elicit experience from a finished project. (using a controlled experiment)
- Study5: A literature study (using systematic review)

3.2.1 Study 1, 2 and 3: Industrial Studies

During the work of this thesis we were involved in three industrial settings. Referred to as study 1, 2 and 3. All these companies were part of the SPIKE project, and as such close cooperation on scientific and company goals were expected. We decided to apply action research as our research method in order to achieve mutual benefits between the practitioners and researchers. In order to produce valid research we followed the five principles of canonical action research suggested by Davison. Due to our involvement in the SPIKE research project, the first principle of Davison was already fulfilled through a company-researcher improvement and research plan that both parties had agreed on. Using the categorization of Avison for control structures in action research, we categorize the different research projects as outlined in Table 7.

Table 7: Categorization of action research control structures

	Study1	**Study2**	**Study3**
Initiation	Collaborative	Collaborative	Collaborative
Authority	Staged Company -> researchers	Staged Researcher -> company	Client
Formalization	Evolved Formal -> Informal	Formal	Informal

We now briefly describe the setting for each company, why they were chosen for this study and what research themes they have been contributing to. We also provide a brief outline of the timeline and how their priorities changed during the period we were involved with them. The companies' priorities influenced the choice of technologies we studied. But as mentioned earlier our viewpoint was always how we could improve the software process from a knowledge management perspective.

Study 1: Mentoring

The company in study 1 is a small software consultancy company, employing 50 people, 30 at their main office in Trondheim and 20 at a branch office, located in Oslo.

Their main source of income comes from three different activities: hiring out developers for pure software development, developing complete solutions for customers and renting out senior personnel as strategic advisors in project management. They have concentrated their customer profile to the domains of healthcare, energy, trade and industry.

Our reason for looking closer at this company was that they expressed an interest in "improving internal knowledge management through revised work processes and internal training of employees in new processes". Particular for this company, was that they were very interested in the human aspects of knowledge sharing, not just codifying the knowledge. As such it fitted nicely into our category for research theme RT3.

The first technology selected by the company was mentoring of employees. We performed interviews to assess the attitudes towards this program in the company and held workshops on how to improve it. The results from our initial assessment are presented in paper P3. Our plan with this result was to follow the implementation in concrete projects, but this turned out to be harder than expected since the company could not come up with any fitting projects. As we were waiting for a suitable project to come up, we started looking at how the company did requirements engineering. We also produced the paper SP4 which suggested an experimental method for improving this process, but as with the mentoring we did not get a suitable project since the project managers were skeptical of trying out new methods. Around this time the company merged with another company, and our main contact person moved to another company. Some attempts were made at studying the knowledge sharing in conjunction with the fusion, but in time the contact between the company and researchers ground to a halt.

Study 2: RUP

The company from study2, is mainly developing software systems with heavy back-end logic and often with a web front-end, typically portals. However, they also develop lighter solutions with most emphasis on the front-end. The company acts as an independent software supplier, though there are close relationships to the biggest customers. Of the 50 employees today, 35 are working as software developers. Java and J2EE are used as development platform. The domain of which the company develops software is mainly for the banking and finance sector, as well as for public sector. The company has run 50 development projects within the bank and finance sector the last twelve years, and about 30-40 projects within the public sector the last 15 years.

Four employees are certified RUP-mentors acting as advisors in other software-organizations, in addition to this they run training courses in RUP and related subjects. The company utilizes their high competence in RUP and most projects are more or less inspired by RUP, however, the company's management saw a need and a possibility to improve their use of RUP by adapting and codifying their development process to the RUP framework. This is what sparked our interest in the company since it fitted nicely with our research theme RT2.

The company wanted to adapt the rational unified process to their projects. Our first intervention was to help the company define their project types. We then held several workshops trying to adapt RUP from a top down perspective but it was soon evident that we had to rethink this strategy. The company then held a series of smaller workshops where the researchers were just observers, and more people of the company were involved. This resulted in an initial version of their downscaled process, and was reported in paper P2. The use of this process was observed in a project and contrasted by other companies and research literature in paper P5. We would have liked to follow more projects, but as with study 1, the company had problems delivering concrete cases and projects we could follow as researchers.

Study 3: Process Workshops

The company in study 3, is a small software consulting company with 20 employees. Their main activities are hiring out consultants as developers, developing complete solutions for customers, and hiring out consultants and project managers as advisors for selecting technology, strategy or process. Typically, no more than four to five consultants are at any time working for the same customer.

One of the identified stumbling blocks for experience sharing and reuse was the lack of a common process and a common set of document templates. In order to remove, or at least reduce this problem, the company wanted to define, document and implement a framework that could be used for development, consultancy and operation. The framework should be easily accessible for all employees and should help them to do their jobs better than today and to show themselves as a highly competent consultancy company. Initially the company planned to downscale the RUP process.

Originally this company was included because of its close fit with research theme 2, and as a possible contrasts case to study2. But after approximately six months the company started to drift away from their original goal of creating a new process and decided instead to document how they worked now. A shift from prescriptive to descriptive modeling. This shift leads to the company covering both research theme 2 and 3.

We started the research in study 3 much in the same way as study 2. The company had the idea to downscale RUP and use it in an electronic process guide. The initial research was thus focused on the electronic guide and what the developers wanted from such a tool. This was presented in paper P1. However, the idea of downscaling RUP never quite caught on and after six months the focus shifted towards documenting their actual process. Inspired by a method developed and used in another SPIKE company, we decided to apply the same method for this company to contrast it in a different context. This process was longer than we had anticipated since the company was rather small and had problems setting aside enough time for the proposed workshops. In the end the process was captured using the proposed method and the study was contrasted to the study where the original method was tested. This was reported in paper P4

3.2.2 Study 4: Controlled Experiment on PMA

This study began quite informally. In our studies of companies in SPIKE, we sometimes used a method for retrospective analysis, called the post mortem analysis. The method is designed to extract important positive and negative experiences from finished projects, or projects at phase transitions and analyze the key causes of these experiences in order to improve future projects. However, we experienced that participation was high during initial phases and dwindled towards the end. During discussions with other researchers who used the same method we got the impression that this was a recurring trend with the method. Based on this we came up with an alteration to the method that we hoped would increase the level of participation.

We tested this new method on a group of students involved in a software architecture course in the spring of 2005. The previous year the students of that year's course had used the original method to analyze their project. When we made a comparison between the methods used in the two years there was a large difference in originality of the ideas.

However, these retrospective sessions were not planned intentionally as a research study and we did not have control of factors that could affect the results (different students, different introductions to the method etc.). On the basis of our experiences from the retrospective sessions in 2004 and 2005, we planned a controlled experiment and performed this in 2006. The motivation for this experiment was to limit other factors that could ruin the experimental results, such as randomization of subjects, different introductions to the two PMA methods, and different working conditions and time limits for the groups. The goal was to get statistical valid data that would support our alteration of the method. The results of this experiment are presented in paper P6.

3.2.3 Study 5: Systematic Literature Review

From the initial studies at the different companies, it was clear that there were many approaches to managing knowledge in a software company. Although we made literature studies into the related areas of each industrial project, and thus gained insight into research theme 1, we found that we were lacking the big picture. We were also unable to find any good survey papers that properly covered the field and that was up to date. We therefore decided to make a systematic review on knowledge management in software engineering. The choice of making the review systematic rather than a regular review, was inspired by the trend of Evidence Based Software Engineering (Dybå, 2005).

The major reason for choosing systematic review over a regular literature review was the desire to get results that was replicable and the possibility to assess the completeness of the search. In order to meet deadlines and keep the workload manageable we needed to limit our search somewhat. However, these limitations are clearly described and our work forms an overview that can serve as a platform if researchers want to expand the overview of the field.

Our search strategy yielded an initial set of 2102 papers, we refined our selection through several steps. In the end, we ended up with 59 papers on knowledge management in software engineering in an industry context. These papers where then categorized in a framework, separating the papers according to approach to knowledge management, and scientific rigor. The papers within each approach were then analyzed with regards to theme and scientific method in order to answer our research questions.

4 Results

We now present the main result from the different studies. We first give an overview of each of the studies, and their contributions. We then sum up the contributions to this thesis.

4.1 Summary of the individual studies

We sum up the major results from each study, we also relate the study to our papers, research questions and contributions. For each study we also present the abstract of the paper(s) reporting the results.

4.1.1 Study 1: Mentoring

The results from this study was the investigation of a mentor program, presented in paper P3, and a research proposal presented in paper SP4 and SP5. However, for this thesis we have chosen to only focus on our study of the mentor program.

The study gave rise to and answered our research question RQ3.1, on how to improve knowledge transfer through mentor programs. We also claim the lessons learned from this study on how to improve a mentor program as one of our contributions in this thesis, C5.

Our main result from this study was that by using the theories from chapter 2.4 we were able to identify the learning taking place as mostly single looped, and consequently suggested methods to remedy this. Also of interest were the identification of several communities of practice where mentoring were taking place in more unofficial manners.

Paper P3 contains several suggestions for improvements that were taken into the revised company program, including: Clear role definitions, possibility of group mentoring, and stimulation of discussion by not giving outright solutions but rather examples of how things had been done.

P3: A study of a Mentoring Program for Knowledge Transfer in a Small Software Company

Mentor programs are important mechanisms that serve functions such as career development as well as knowledge transfer. Many see mentor programs as an efficient,

inexpensive, flexible and tailored way of transferring technical knowledge from experts to less experienced employees. We have investigated how a mentor program works in a small software consultancy company, and propose that the learning effect of the program could be improved by introducing methods to increase the employees' level of reflection.

4.1.2 Study 2: Rational Unified Process Tailoring

This study followed a medium sized consulting company as they tried to downscale and tailor the rational unified process for use in their software development projects. The results are published in paper P2, P5 and SP6.

Our specific research question for our study of the rational unified process was the research question RQ2.2: What are the limitations, benefits, prerequisites and cost of tailoring and introducing the rational unified process? And the study is also the source of our claim C2, tailoring the Rational Unified Process.

When we started this study, the company already had some experience from using the rational unified process in a free form, with no restrictions or guidelines placed on the employees, their previous experiences are reported in paper SP6.

Our initial work with the company, reported in P2, showed us that a top-down approach was heavily dependent on expertise, and in the end did not work very well for the company. A second attempt, using a bottom-up approach and involving as many of the employees of the company as possible, created a much better result. A key factor from our viewpoint was the focus of the workshop that kept the discussions on track and delivered good codifiable results.

In order to extend our results and to achieve better external validity, we compared the results from this study with two other studies done in the SPIKE context, also investigating companies using the rational unified process. We also extended our results with a systematic review of the research literature. This extension of our original study was reported in paper P5. The main result here was that despite the apparent interest in the rational unified process, little empirical research had been done on implementations and downscaling. The results, however, confirmed our belief that the process in itself is too complex, dependent on experts and needs to be simplified.

P2: Tailoring RUP to a defined project type: A case study

The Unified Process is a widely used process framework for software development. The framework is covering many of the roles, activities and artifacts needed in a software development project. However, a tailoring of the framework is necessary to fit specific needs. This tailoring may be accomplished in various ways. In this paper we describe a concrete attempt to tailor the Rational Unified Process to a defined project type; a Mainstream Software Development Project Type. The paper has focus on the process of creating the tailored Rational Unified Process as well as the resulting Rational Unified Process. The paper makes some conclusions and has a proposition for further research.

P5:Tailoring and introduction of the Rational Unified Process

RUP is a comprehensive software development process framework that has gained a lot of interest by the industry. One major challenge taking RUP into use is to tailor it to specific needs. This study presents a review and a systematic assembly of existing studies. We have found that tailoring RUP is a considerable challenge by itself and that tendency is turning from large complete process frameworks towards smaller and more light-weight processes.

4.1.3 Study 3: Process Workshops

This study followed a small sized software consulting company. Their original goal was to create an electronic process guide, but after some time the focus changed to using process workshops to define their software process. The initial attempts at creating an electronic process guide is reported in paper P1, the results from using the process workshops coupled with the results of using process workshops in another SPIKE study is presented in paper P4.

Our initial work in this study gave rise to research question RQ2.1: What do developers want from a knowledge sharing tool? This gave us input on what they viewed as important artifacts for codification, like templates, patterns and process models.

The really interesting results, however, stem from the later stages of this study, when the focus changed to defining the software process through process workshops. By applying a method developed and tested in another SPIKE study, we had the chance of contextualizing and improving the method. This lead to our research question RQ2.3 which deals with how the context affects the final artifacts of the method, and RQ 3.2 which deals with how the context affects the execution of the method. Through this study we make claim C3, that we improved the method by contextualizing it.

We found that the discussions and results of the workshops were greatly influenced by the context of the companies, and concluded that previous process experience was a major factor influencing the results. We were also able to point out where prescriptive modeling was better than descriptive modeling. And whether to focus on artifacts or activities. We also confirmed earlier observations that the method was a good approach to company learning, and that user involvement had an impact on process uptake and use.

P1: Harvesting Knowledge through a Method Framework in an Electronic Process Guide

A key leverage for small software consultancy companies is the collective knowledge possessed by their consultants. There have been some studies in the literature on how to harvest and transfer this knowledge, but most studies are aimed at large multinational corporations. In this paper we describe an ongoing research project, aimed at improving knowledge sharing in a small software consultancy company through the use of a method framework in an electronic process guide coupled with an experience repository.

P4: Defining Software Processes Through Process Workshops: A Multicase Study

We present the application of the process workshop method to define revised work processes in software development companies. Through two empirical action research studies, we study the impact of company premises and goals on the execution and subsequently on the results of the method. We conclude that both premises and goals will influence the workshops, and suggest how the focus of the workshops should be altered to achieve better results depending on the context. We also strengthen previous claims that the process workshops are a good arena that fosters discussion and organizational learning, and that involvement in the workshops leads to higher acceptance and usage of the resulting process.

4.1.4 Study 4: Post Mortem Analysis

This study began quite informally. In our studies of companies in SPIKE, we sometimes used a method for retrospective analysis, called the post mortem analysis. The method is designed to extract important positive and negative experiences from finished projects, or projects at key transition stages and analyze the key causes of these experiences in order to improve future projects. We did however notice that not all parts of this method was working equally well.

In order to remedy this, we suggested some improvements to the method and tested it on a group of students. The results were not as expected, as the new method produced far fewer causes. However, looking at the data qualitatively suggested that the causes were of a higher quality than the causes suggested by the old method.

Seeking to confirm these data, we planned a controlled experiment were we sought to control the confounding factors. The controlled experiment confirmed our earlier indications, and is reported in paper P6, which answers research question RQ3.3, improving knowledge sharing through project retrospectives. The study gives rise to claim C4, improvement of the PMA. The major benefit of the revised method is that the new method is less dependent on a professional facilitator, and as such much better suited to context where experienced facilitators are not readily available.

P6: Improving the Effectiveness of Root Cause Analysis in a Retrospective Method: a Controlled Experiment

Retrospective analysis is a way to share knowledge following the completion of a project or major milestone. However, in the busy workday of a software project, there is rarely time for such reviews and there is a need for effective methods that will yield good results quickly without the need for external consultants or experts. Building on an existing method for retrospective analysis and theories of group involvement, we propose improvements to the root cause analysis phase of a lightweight retrospective analysis method known as Post Mortem Analysis (PMA). In particular, to facilitate brainstorming during the root cause analysis phase of the PMA, we propose certain processual changes to facilitate more active individual participation and the use of less rigidly structured diagrams. We conducted a controlled experiment to compare this new variation of the method with the existing one, and conclude that in our setting of small software teams with no access to an experienced facilitator, the new variation is more effective when it comes to identifying possible root causes of problems and successes. The modified method also produced more specific starting points for improving the software development process.

4.1.5 Study 5: Systematic Review

This study happened in parallel to the previously reported studies, and was an ongoing research for most of the time of this PhD study. The results is reported in paper P7. The study gives answers to research question RQ1.1, on the themes of previous research, and to RQ1.2, about methods applied within the field. The study is the source of our claim C1, an overview of the field.

Our study reports on previous research on knowledge management in software engineering. A general finding was an increased interest in the field, picking up from 1999 and onwards. Another general finding was a trend to move from lessons learned oriented to more systematic empirical studies, see Figure 5.

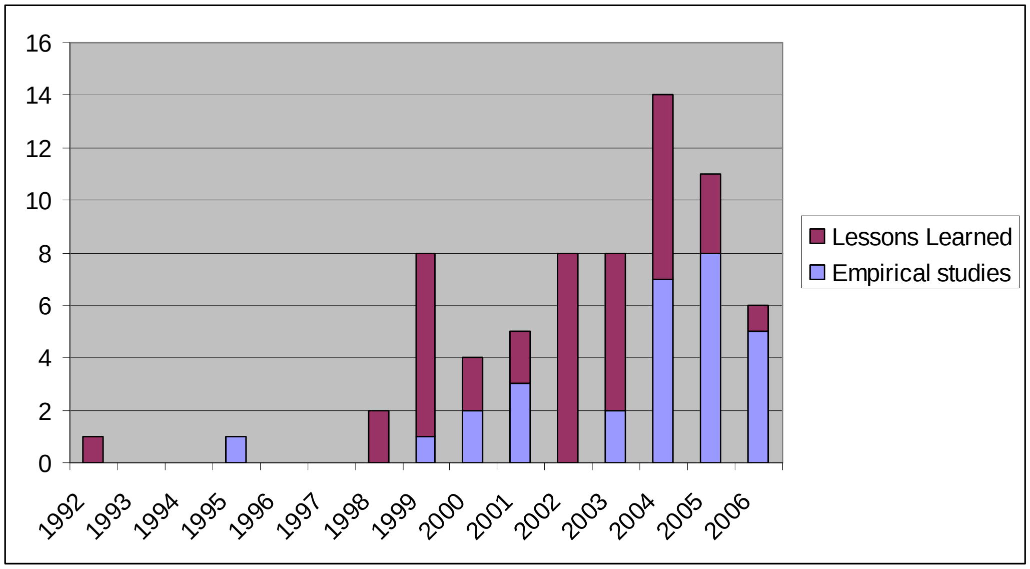

Figure 5: Studies of KM in SE

Using the classification scheme of Glass described in section 2.3 we found the field leaning heavily towards the technocratic side, with only some papers published on the behavioral aspects (see Table 8). Within the technocratic aspects, the focus seemed to be on the engineering and systems schools. The economic, spatial and cartographic schools were found underrepresented in the field. We also found very little overlap between the concepts being studied within each school.

Table 8: Categorized papers

	Systems	Cartographic	Engineering	Commercial	Organizational	Spatial	Strategic	SUM
Empirical studies	6	1	12	0	3	0	3	25
% distribution, empiricla studies	*24*	*4*	*48*	*0*	*12*	*0*	*12*	100
Lessons learned reports	20	0	9	0	2	1	9	41
% distribution, lessons learned reports	*49*	*0*	*22*	*0*	*5*	*2*	*22*	100

Concerning research methods, the majority of the identified papers were lessons learned reports and not scientific studies. Of the empirical studies more than half of them were case studies, but we also found some field studies, action research studies, ethnographic studies and one laboratory experiment. See Table 9.

Table 9: Research methods for KM in SE

	Action Research	Case study	Etnography	Experiment	Field study	Sum
Systems	1	3	1		1	6
Cartographic			1			1
Engineering	1	8		1	2	12
Organizational		3				3
Strategic	1				2	3
Sum	3	14	2	1	5	25
%	12	56	8	4	20	100

P7: Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts and Research Methods Used

Software engineering is knowledge-intensive work, and how to manage software engineering knowledge has received much attention. This systematic review identifies empirical studies of knowledge management initiatives in software engineering, and discusses the concepts studied and the research methods used. Seven hundred and sixty-two articles were identified, of which 68 were studies in an industry context. Of these, 29 were empirical studies and 39 reports of lessons learned.
The majority of empirical studies relate to technocratic and behavioral aspects of knowledge management, while there are few studies relating to economic, spatial and cartographic approaches. More than half of the empirical studies were case studies.

4.2 Overview of Contributions

The identified contributions as described in chapter 1.6 relates closely to the studies we carried out with one major contribution from each one. We will briefly describe the practical details of each contribution here before moving on to the discussion part and linking the contributions with the overall themes and theories.

C1: An overview of the research literature on empirical studies of knowledge management in software engineering.
We identified 29 empirical studies and 39 lessons learned reports. These papers were mainly concentrated within the technocratic schools where the emphasis was on the engineering and systems school. Where research methods were applied this was mainly case studies. For a short summary of the major findings of this contribution, see Chapter 4.1.5. This contribution is described in detail in paper P7.

C2: A method for tailoring the Rational Unified Process to the development process of a software consulting company.
In general it seemed that the adaptation was best done as a simple, pragmatic process not as a heavily up-front planned and strictly managed process. Our approach involved a series of workshops in which the key success factor seemed to be focus. Focus both through a specific project type, specific process elements and through phases or disciplines. Another key success factor was that a workshop consisted of persons with the proper experience with regards to the focus. The focus on a specific project type seems to have kept the participants on track throughout the adaptation process. It seemed to have eased the process since everyone had a clear concept of what should be done in that particular project type. This contribution is described in detail in paper P2 and expanded upon in paper P5.

C3: Improvements of the Process Workshops method by contextualization.
We found that the major influencing factor on the execution of the workshop was the developers' previous process experience. If the employees of a company were used to working according to a process, the workshop would benefit from a focus on prescriptive modeling. If, however, no clear process existed, the focus of the workshops should be on reaching an agreement on the current process before improvement was suggested, or in other words, descriptive modeling. We also found that the method could improve by focusing on the right deliverables. If the process workshop approach was used to reach an agreement on the current process, the main deliverable should be a list of artifacts. However, if the process workshop approach was used to specify future processes based on best practice, the main deliverable should be a list of activities. This contribution is described in detail in paper P4.

C4: Improvement of the root-cause analysis phase of the lightweight Post Mortem Analysis for more effective project retrospectives.
We changed the root-cause analysis phase by altering the brainstorming technique from interactive to more nominal, and introducing the more free form diagrammatic technique of causal mapping. A group is defined as nominal if its members work independently, but in each other's presence. A group is defined as interactive if its members generate ideas in face-to-face discussions. This contribution is described in detail in paper P6.

C5: Proposal of methods to increase the learning effect of mentor programs in software engineering.

The main challenge we identified from the research literature and our interviews, was to increase the level of reflection and move from a single looped learning scheme to more double looped. Several suggestions for improvements were taken into the revised company program, including: Clear role definitions, possibility of group mentoring, and stimulation of discussion by not giving outright solutions but rather examples of how things had been done. This contribution is described in detail in paper P3.

5 Discussion of results

We will now discuss the results of our studies. We return to our three research themes and discuss how our studies have contributed towards these. We will not discuss the concrete research questions, since those discussions have been covered in the individual papers. For discussions on the validity of our contributions we also defer the reader to the enclosed papers.

For each research theme we discuss which of our contributions have had an impact on it. We relate these contributions to the state-of-the-art, both showing how they fit with existing literature, and how they have extended the field. For RT2 and RT3 we make a theoretical comparison on the learning effects of the different contributions, using theories of organizational learning to point out similarities and differences. In addition we reflect on each theme to present what we feel is the major contributions within the themes towards the overall goal of the thesis.

Finally, we add some reflections on our research context, the SPIKE project.

5.1 Research on knowledge management in software engineering (RT1)

Our first research theme was chosen to give us an overview of the research literature in the field we chose to study. Both in order to find out what had previously been done, but also to have a framework to compare our own work against.

Our major contribution towards RT1, is C1, the overview on the field. We chose the systematic review approach for this contribution. In retrospect we see that this method consumed significantly more effort than we first expected, and we had to make tradeoffs to limit the scope of the search and presentation. We could of course have asked more questions within this contribution. What theories have been used? How effective are the different approaches? What settings have the methods been tested in? However, we were faced with time and resource limitations, and we had to make a choice. We believe that the two questions we asked (RQ1.1 and RQ1.2), helps to establish a baseline for the field. We know what has been investigated empirically, and we know how.

Contribution in relation to state-of-the-art

Comparing our findings for this research theme with the state-of-the-art presented in chapter 2, confirms the results of (Hansen, 2004) on the distribution of research in the SPI field. Our study excluded what Hansen et al. call prescriptive studies. But based on the number of exclusions and the criteria for these, we found that the field is biased towards the prescriptive and descriptive studies with very few reflective contributions.

According to the MAP framework of Aaen (2001), which provides a framework for categorizing SPI studies, research in SPI as a whole has focused primarily on aspects related to norms, and compliance to these. Although we haven't specifically categorized the final selection against the MAP framework, the focus of our selection seems not to be as heavily biased towards norms for our field as Aaen et al. suggests that it is for the SPI field as a whole. Our study do however address an area which Aaen et al. points out as an area needing more research, namely the role of competence in SPI and how to distribute this competence.

From the knowledge management perspective, using the theories suggested by Hansen (1999) we observe a heavy bias towards the codification strategy. We also echo the findings of Christensen (2005) that the majority of studies seem to focus on the creation and transfer of knowledge. The classification scheme suggested by Earl (2001) proved useful for categorizing studies of knowledge management in software engineering, although a few studies could not be classified in the framework. In Earl's framework we saw a clear bias towards the technocratic approaches.

How then can we claim to have contributed towards the state-of-the-art? Some attempts have previously been made to establish an overview of knowledge management in software engineering. Rus et al. (2001), Lindvall et al. (2001), and Dingsøyr and Conradi (2002). However, these studies are from 2001/2002 and none of them are specifically targeted to establish a complete and systematic overview of all studies in the field. We therefore believe that our contribution is a good building block for establishing a complete and systematic overview of scientific studies within the field.

Reflections on the Research Theme

By using our contribution as a possible building block for the field, researchers who are interested in doing research within this field can use our study to identify what papers have been published with relations to the concepts they want to investigate, and use it as a starting point for their own research. It is also possible by looking at our results to identify clear holes in both what has been investigated and how it has been investigated. In addition we were able to use the framework from this contribution to relate our other contributions to the field as a whole.

It is interesting to look at our own contributions in light of the framework we used to categorize the field in C1. Most of our contributions C2, C3, and C4 can be placed within the engineering school. This is not surprising since the foundation of this school is the processes which ties closely with our focus on SPI. The engineering school was identified as having the largest amount of empirical studies within all the schools. Yet when we looked closer at the studies there were little to none overlap between them. As such the school needs more attention to fill in the gaps, and we believe our contributions C2, C3 and C4 contributes towards that. Our contribution C5 can be categorized in the organizational school, an area clearly lacking attention in the field. However, getting data proved troublesome and thus the contribution is not as well investigated as we had hoped for.

5.2 Application of knowledge management to improve the software process through codification of knowledge (RT2)

Our second research theme, as well as our third was chosen with a basis in the theories of Hansen (1999), which states that there are basically two approaches to knowledge management. RT2, deals with the first of these, codification. There were several companies in SPIKE interested in improving their development processes by codifying them. We chose two cases that had contrasting methods to define their development process: One using the Rational Unified Process approach, which takes a top down approach and the other using the Process Workshops that takes a bottom up approach. These studies resulted in contribution C2 and C3 respectively.

The two investigated methods can only be said to represent a fraction of this research theme. However, taken together they do give an important insight into a common starting point for SPI, the codification of processes into process frameworks. Before you can improve your process, it is important to know what process the organization is following. Also worth noting, is that despite their different starting points, they evolved into using almost identical knowledge generating processes seen from the theoretical perspective. Our findings from the studies of both these methods support earlier research that involvement of developers in defining the process is the key to creating ownership and subsequently use of the resulting processes (Lawler, 1987; Guzzo, 1996; Vandenberg, 1999; Riordan, 2005).

Contribution in relation to state-of-the-art

According to Hansen (2004) our contributions can be seen as descriptive/reflective, which contributes towards the SPI field which is heavily biased towards prescriptive studies. In the MAP framework of Aaen (2001), we have made contributions towards the competence factor. In the field of KM this research theme obviously contributes towards Hansen’s codification strategies, particularly the school of engineering according to the framework of Earl (2001).

Concerning state-of-the-art related to RUP, we performed a literature review to find related studies, see P5 for details. Our findings were that despite indisputable interest in the subject, the total amount of empirical studies on the adoption and introduction of RUP was surprisingly low. This means that our empirical studies on the adoption and introduction of RUP, P2, SP6 and P5, has contributed significantly to the state-of-the-art of this subject.

In a systematic review of software process tailoring by Pedreia (2007), our study reported in P2 is identified as one of 28 primary studies, from an initial selection of 394. Method tailoring covers all method adaptations, not only RUP. Our study seems to have made an impact on the study of method tailoring in small companies (only 20% of the studies were in small companies the other 80% was from large), company level tailoring as opposed to project level tailoring (35% vs. 65%), and on informal approaches (33% vs. 67%).

The published empirical material concerning the Process Workshops method was limited to the original introduction of the method (Dingsoyr, 2005) and a Finnish application of the same method (Pikkarainen, 2005). Our study extended the results from the original introduction, by comparing it to a contrast case. Through this we were able to improve the method based on the context it is applied to. Our contribution to the state-of-the-art of this method can thus be said to be significant.

Contributions in Relation to Organizational Learning

Using the theories of organizational learning outlined in chapter 2, we can make a theoretical comparison between the two studies leading to contribution C2 and C3. For both companies, their initial attempts can be described as combinatory, trying to take an explicit process and making a new explicit process. For both studies we found that such a process depended heavily on an expert and the resulting process would be too far from the regular process unless more employees were involved in defining it. Both companies then chose to define their process through a more bottom up approach. The company in C2 held their own workshops, using the RUP framework as guidelines, while the company in C3 tried the process workshop approach suggested by the researchers.

Comparing the two cases, we find that the workshops take on a different meaning in terms of learning. The company where the developers were accustomed to working with process definitions were able refine their tacit process knowledge into external activities. The employees who were not used to working with a defined process used the workshops as primarily socializing their knowledge and reaching a consensus. This resulted in an agreement on artifacts, rather than activities. We could also observe the importance of having a medium for collecting the externalized knowledge. In C2 we observed the flow of knowledge from the individuals to the group levels through workshops and projects and into the final wiki, representing the organizational level. Through C3 we observed the flow from the individuals to the group level in the workshops, but the flow to the organizational level seemed slow or lacking. In addition for C2, we could follow a new process as the project manager got access to the new process and how she internalized this information into operational knowledge.

Looking at the process in the study leading to C2, from the perspective of Wenger's theories, we observed the formation of a community of practice surrounding the process improvement group. As the process was defined, involving as many employees as possible, we saw the community form with the process group as the community's nucleus, with employees in a more peripheral position. Since the workshops were conducted in parallel and early in the process, most employees can be said to be on a peripheral trajectory, and are not necessarily drawn into the community of practice. Our observations of a project using the redefined process allowed us to study the interaction between the community of practice formed around the new project and the community of practice formed around the process. When actively using the redefined process, the project manager can be seen to be on an inbound trajectory to the community of practice. In this case the wiki functioned as the boundary object, allowing contact through reification. The project manager and a member of the process group also had frequent meetings, resulting in contact through participation or brokering. This contact was beneficial for both communities which mutually influenced each other to create an improved process.

In the study leading to C3, however, where the workshops were held in a more serial manner, we observed a community of practice formed around the people participating in the workshops. The core formed from the people most often participating in the workshops, while the peripherals could be considered those only participating in one or two workshops. We observed a larger degree of employees being on inbound trajectories to the community of practice, proportional to the number of workshops they participated in. Developers who did not participate in the workshops, fell outside this community of practice, and it was commented that the new process models had a much larger impact inside the community than outside. The workshops were also organized to fit with the different phases of the development process. We found it useful to invite participants with particular knowledge of the individual phases, or coming from a theory perspective, communities formed around the different phases of the process. Since the company lacked a tool to implement the results from the workshops it was up to the researchers to create boundary objects from the workshop sessions.

Reflections on the Research Theme

In our view, the most interesting aspect from our study of codification approaches in software process improvement is the balance between artifacts and activities. From our workshops leading to both C2 and C3, a key success factor was to have a specific focus throughout the session. As we discovered through C2, one thing is the focus on a specific project type, specific process elements and specific phases or disciplines. But as C3 showed us, and which we in retrospect can corroborate with our observations on C2, another important focus is whether the focus is on codification of artifacts or activities. Whether the focus should be on artifacts or activities, seems to be related to the process maturity of the company. If the company has some experience with defined processes the focus can be directed towards activities. If there is little or varied experience with process compliance, it is better to direct the focus towards the artifacts. This observation is, however, is built on a few case studies and as such we should be careful about generalizing to the entire software engineering field. It is however an interesting proposition for further research.

Another important finding from both our contributions towards codification is that developers do not inherently trust or adhere to what is written in process guides, and as such process conformance will be low unless measures are taken to prevent this. Both our studies showed that including developers in the creation of the new process creates ownership and future process compliance. Another measure is coupling the process steps with a rationale for complying with it. In C2, the company added motivating texts explaining why they felt each step was necessary and possible risks by omitting it. In C3, they wanted to couple each step with an experience base, so the developers could see what experience other developers had with using the steps of the process.

5.3 Application of knowledge management to improve the software process through personalization of knowledge (RT3)

The third research theme, like the second stems from Hansen’s strategies for managing knowledge. It relates to our belief that a company’s software practices are ultimately based on the knowledge and skills of its employees. A lot of process knowledge is embedded in the heads of employees in software companies, and so if we increase the sharing of knowledge and experience between the employees we will improve the overall process too. We chose to study three methods for knowledge sharing in software companies: process workshops, the post mortem analysis, and a mentor program. Resulting in contribution C3, C4 and C5 respectively. Although contribution C3 was included in RT2, it is also relevant for RT3 due to the amount of tacit knowledge sharing we observed during the workshops.

As with research theme RT2, we had to narrow our focus with the studies and research questions. We have gained insight into the concrete methods we chose to study, and we also believe that our observations provide valuable general information to the research theme. The methods we have studied are only a small part of the methods that could possibly contribute to this research theme. However, a general lesson from these studies is that when theories of knowledge management are applied within a software engineering setting, we are able to increase the level of knowledge sharing.

Contributions in relation to state-of-the-art

Like RT2, research theme RT3 also contributes towards the competence factor in the MAP framework of Aaen (2001). Concerning the overview by Hansen (2004), both contribution C3 and C4 can be considered descriptive/reflective. Due to lack of data, however, C5 must be considered mostly prescriptive although it does contain descriptive data and reflection on theory.

In the field of KM, this theme can be seen to address one of the main concerns of Alavi (2001): How to facilitate the flow of knowledge between individuals so the maximum amount of transfer occurs (assuming that the knowledge individuals create has value and can improve performance). In the framework of Earl (2001) we place the contributions of this theme in both the engineering- and the organizational school.

As we discussed under RT2, the material published on the process workshops method was limited, and our results extend the results from the original introduction of the method. In the framework used in C1, we classify the study as belonging to the engineering school.

Concerning contribution C4, there is a wide variety of methods for conducting project retrospectives. The lightweight postmortem analysis we chose as a starting point is but one of these. As such, our improvement to this method is hardly a large impact on the field, but the general theories we based the improvements on, should be applicable to other methods as well. Related to C1, this study can also be placed in the engineering school. It expands on some of the papers relating to project retrospectives and improving the software process based on the experience from finished projects.

Our final contribution towards this research theme is C5, mentoring. In the research literature we found plenty of studies on mentor programs, but no such studies in a software engineering setting. Our study is not groundbreaking within the field, but it is a contribution towards contextualizing mentoring in a software setting. Related to our framework in C1, we can categorize this contribution within the organizational school. As such it contributes towards the more behavioral aspects of KM strategies, an area we found to be underrepresented in our overview.

Contributions in relation to Organizational Learning

From the perspective of Nonaka and Takeuchi's theories, the most prevalent form of knowledge conversion in the study leading to C3, the process workshops, was socialization, resulting in sympathized knowledge. A whiteboard and yellow stickers were used to capture this knowledge in explicit form, resulting in conceptual knowledge. During this process we observed good knowledge flows from the individual to the group level. This was also commented on by the participants themselves, who claimed the workshops to be the best action towards learning in the company that year.

For the study leading to C4, the post mortem analysis, we observed a similar effect: The primary mean of knowledge sharing for this method was socialization, ideas were generated primarily through discussions and brainstorming. A whiteboard served as a secondary tool allowing the group to externalize their thoughts into conceptual knowledge. For this method we also observed good knowledge flows from the individual to the group level.

From the viewpoint of Nonaka and Takeuchi, the sharing of knowledge in a mentor program, our contribution C5 is inherently socialization. However, with no actual data from the program in practice, it does not make sense to delve deeper into this theory here. Instead we will draw upon another theory, namely Argyris & Schön, which we applied in creating the revised program. During our interviews we drew upon this theory and classified the actual learning as single looped. By increasing the level of reflectiveness or in the words of the theory, making the learning double looped, the company should be able to increase the learning effect of the mentor program.

From Wenger’s perspective, a community of practice formed around the people participating in the workshops in the study leading to C3. The core formed from the people most often participating in the workshops, while the peripherals could be considered those only participating in one or two workshops. The developers not participating in the workshops fell outside this community of practice. It was commented that the new process models had a much larger impact inside the community than outside.

In the improved post mortem analysis in C4, Wenger’s theories offer little insight into the process, as the knowledge is generated within the project group, serving as a community of practice. The brainstorm can be seen as participation, while the whiteboard serves the need for reification. The final report then serves as the boundary object from the development group. In the mentor program leading to C5, there is little to be gained from Wenger’s theories. We are talking of brokering between communities of practice. But as stated above we did not get the chance to study this in practice. We did however use Wenger’s theory to identify several communities of practice where mentoring were taking place in an unofficial manner.

Reflections on the Research Theme

Our systematic review of the field, C1, showed a clear bias towards codification approaches to knowledge management in software engineering. Personalization, however, is an area that is not widely covered in the research literature on software engineering. Even the technocratic school that supports a personalization approach, the cartographic school, is poorly represented in empirical research. Although we have made some contributions towards this theme, much research remains.

However, as our studies have shown, it is not only the research field that leans towards codification approaches. Companies also tend to go for a solution with more tangible results, and ignoring the more human aspects and direct transfer of tacit knowledge.

While our focus in RT2 can be characterized as looking at knowledge flows from the group level to the organizational level, RT3, can be said to be focusing more on knowledge flows between individuals and from the individual to group level. The individual learning effect also seems to be greater in these initiatives. This underlines the need for further research into this aspect of knowledge management. We believe that the software engineering field can improve by taking a more human-centric view at our activities, thus reducing the tool focus which is currently dominating the field.

Our main contribution towards RT3, concerns the way we structure the brainstorming in our workshops. We believe that our strategy of letting the participants work nominally at first, and then structure their discussion interactively around simple tools like a whiteboard with post-it notes, have contributed towards good learning effects. This is mainly seen from our experiment yielding C4, but it is also observed in the workshops conducted as part of C3 and C5. This result, add to the evidence in organizational psychology where it is claimed that nominal groups outperform interactive groups in a brainstorm.

5.4 Reflections on the research context: the SPIKE project

In Software Engineering (SE), as well as in Information Systems, empirical studies of real industrial processes and products are necessary to identify relevant research agendas. However, research focus and research activities can vary videly, from collection of "what-is" data, to ambitious and technology-driven "tool" projects, and to company-wide SPI programs. Industrial organization, technologies, markets and products are also in a continuous flux with little concern for stable researcher agendas.

All this means, that academic SE research in industry carries a substantial risk of yearly cancellation or refocusing of agreed-upon, mutual activities. To be able to stick to a 4 year "grand" research plan between industry and academia is rare or downright impossible. So we have to live with, in our case, smaller studies that inevitably are more heterogeneous and less controlled than desired. This is simply the prize of industrial relevance, and indeed of any cooparation at all. We believe that our approach towards the industrial partners, using action research, compensated somewhat, but not fully for this conflict of interest.

Luckily, we had a larger and relative long-term (3-4 years) cooperation project context (SPIKE) to participate in, where the choice of interesting partners varied. The industrial partners were also sympathetic to academic cooperation. Finally, there were 8-10 researchers from three research institutions regularly working on the project and usually in pairs towards the same company. All this softened the traditional conflicts mentioned above.

Also, as we outlined in chapter 3, having a guiding theme for the thesis proved invaluable in this setting where company strategies frequently changed to adapt to the realities of the world. Having the overall theme of looking at SPI from the viewpoint of KM, allowed us to adapt our research to the company preferences without deviating too far from our original directions.

6 *Conclusion*

Through this thesis we have explored and reported research related to improving software process through use of knowledge management. The research which we have carried out throughout this thesis has provided valuable insights into three main research themes, and resulted in five major contributions. We now sum up our main findings and outline possible future works based on our results.

6.1 Knowledge management in software engineering

Our first research theme investigated the previous research in the field of knowledge management in software engineering. We have one major contribution in this theme:

C1: An overview of the research literature on empirical studies of knowledge management in software engineering. Through a systematic review we created an overview of the research literature to identify what had been investigated and where the holes in the field were. We found a clear bias towards codification strategies, and the technocratic approaches. We also found a bias towards prescriptive and descriptive studies with few reflective contributions. Comparing the rest of our studies to the framework we used for categorizing the field, we claim to have made significant contributions to the engineering school of knowledge management.

6.2 Codification strategies

Our second research theme investigated the codification strategy on knowledge management in software engineering. We focused on two specific methods for codification within this theme. This lead to contributions in the form of two improved methods for eliciting and describing the software process of a company:

C2: A method for tailoring the Rational Unified Process to the development process of a software consulting company. Through an action research study we gained insight into the process of tailoring the Rational Unified Process to the development process of a medium sized software company. Our results were contrasted and strengthened by two other case studies and a systematic literature study. Through this contribution, we realized the importance of focus to properly steer workshops to achieve satisfactory results. We also concluded that the RUP is inherently too heavy and dependent on experts to achieve proper tailoring.

C3: Improvements of the Process Workshops method by contextualization. Through two action research studies where the Process Workshops method was applied to define the software process for two companies, we gained a deeper understanding of how the company context affected the results of this method. Our main finding related to codification was that the level of process maturity in the organizations affected the workshops and consequently the final documented results. In an organization with sufficient process maturity they produced lists and descriptions of activities, while in the organization where the process maturity was lower, the discussions turned towards producing lists and descriptions of artifacts.

6.3 Personalization strategies

Our third research theme investigated the personalization strategy on knowledge management in software engineering. Within this theme we investigated three specific methods for knowledge sharing, leading to three main contributions.

C3: Improvements of the Process Workshops method by contextualization. Through two action research project where the Process Workshops method was applied to define the software process for two companies. In addition to the insights we gained under RT2 by using this method, we gained deeper understanding of how the company context affected the execution and sharing of knowledge during such workshops. Again the process maturity seemed to affect the discussions, allowing the organization with sufficient process maturity to discuss prescriptive modeling, while the organization with low process maturity leaned towards descriptive modeling.

C4: Improvement of the root-cause analysis phase of the lightweight Post Mortem Analysis for more effective project retrospectives. We proposed changes to the brainstorming in the root-cause analysis phase of the post mortem analysis method, which we then tested in a controlled experiment using 4th year masters students. The result was a more effective method in that it discovered deeper and more explicit causes for project problems in the same amount of time as the previous version. We also discovered that the revised method was less dependent on experienced facilitators.

C5: Proposal of methods to increase the learning effect of mentor programs in software engineering. Through an action research project in a medium sized software company, we gained deeper insight into how knowledge was shared in a mentor program, and we proposed several modifications to the program that could increase the learning effect. The most important improvement was to increasing the level of reflection to move the learning cycle from single to double loop.

6.4 Research Goal

Returning, finally, to our research goal for this thesis: *How can Knowledge Management be applied to Software Engineering in order to foster Software Process Improvement?* We found that by taking a knowledge management perspective on software process improvement, we could identify and increase learning effects, a key factor in getting developers to improve their practices. Our studies also showed that most research within software engineering has been directed towards the codification strategies, and that research on transfer of tacit knowledge through personalization is lacking, even though the learning effect on the individual level seemed greater through these. Further, our studies showed that communities of practice sprung up around SPI efforts. Participation in these communities seemed to be the key factor for the impact of the revised processes. A key challenge is to involve and keep the developers in these communities and make sure they don't drift out of them, once their involvement has ended. As we have seen there are many possible applications of knowledge management in software engineering, and we have tested but a few during the work on this thesis. But, as previous researchers have pointed out, there are many possible routes to the goal, and no single approach is necessarily the best for all possible contexts. Our studies have contributed towards the state-of-the-art by contextualizing some methods, but there are still a lot of possibilities for research within the field.

6.5 Future Work

Our three research themes lend themselves nicely to possible future directions for research we have started in this thesis.

Our overview of the field does currently only include studies in industrial contexts, and can be greatly expanded by adding prescriptive studies from academia. There are also possibilities in extracting more information from the studies already identified, concerning contexts and method impact.

Of the two methods we studied within the codification strategy, we would say the process workshops method has the most potential for further studies. As we found in our literature study on the RUP, the industry has started to realize that the process in its original form was indeed too heavy, and is launching new and more agile versions instead. As we also saw in our study of downscaling the RUP, the organization ended up with a simple and pragmatic approach, using workshops to involve developers in defining their process. We believe the process workshops method can offer this simple and pragmatic framework for structuring discussions on software processes. It would also be interesting to follow the indications of process maturity relating to the balance between discussions on artifacts or activities.

As we saw from our overview of the field, the personalization strategies have not received much coverage. It should therefore be interesting to follow this track in future research. From the methods we investigated within this theme, we believe the improved post mortem analysis could yield the most interesting results, if taken further. With agile methods gaining in popularity in software engineering, the need for effective dissemination of experience through short meetings or workshops is definitely needed, and we believe much can be gained by more research on how to conduct project retrospectives in software engineering.

References

Ahonen, J. J., Forsell, M. and Taskinen, S.-K. 2002. A Modest but Practical Software Process Modeling Technique for Software Process Improvement. Software Process Improvement and Practice 7(1): 33-44.

Alavi, M. and Leidner, D. E. 2001. Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. MIS Quarterly 25(1): 107-136.

Argyris, C. 1990. Overcoming Organizational Defences: Facilitating Organizational Learning. Prentice Hall.

Argyris, C. and Schön, D. A. 1996. Organizational Learning II: Theory, Method and Practise. Addison Wesley.

Avison, D., Baskerville, R. and Myers, M. 2001. Controlling Action Research Projects. Information Technology & People 14(1): 28-45.

Avison, D., Lau, F., Myers, M. and Nielsen, P. A. 1999. Action Research. Communication of the ACM 42(1): 94-97.

Aybüke, A., Jeffrey, R., Wohlin, C. and Handzic, M. 2003. Managing Software Engineering Knowledge. Springer Verlag.

Basili, V. R. 1996. The role of experimentation in software engineering: past, current, and Future. 18th International Conference on Software Engineering. 442-449

Basili, V. R. and Caldiera, G. 1995. Improve Software Quality by Reusing Knowledge and Experience. Sloan Management Review 37: 55-64.

Basili, V. R., Caldiera, G., McGarry, F., Pajerski, R., Page, G. and Waligora, S. 1992. The software engineering laboratory: An operational software experience factory. 14th International Conference on Software Engineering. Melbourne, Australia. 370-381

Basili, V. R., Selby, R. W. and Hutchens, D. H. 1986. Experimentation in Software Engineering. IEEE Transaction on Software Engineering 12(7): 733-743.

Baskerville, R. and Pries-Heje, J. 1999. Grounded action research: a method for understanding IT in practice. Accounting, Management and Information Technologies 9(1): 1-23.

Baskerville, R. L. and Wood-Harper, A. T. 1996. A Critical Perspective on Action Research as a Method for Information Systems Research. Journal of Information Technology 11(3): 235-246.

Becker-Kornstaedt, U. 2001. Towards Systematic Knowledge Elicitation for Descriptive Software Process Modeling. Lecture Notes in Computer Science 2188: 312-325.

Bergström, S. and Råberg, L. 2003. Adopting the Rational Unified Process. Addison-Wesley.

Birk, A., Dingsøyr, T. and Stålhane, T. 2002. Postmortem: Never Leave a Project Without it. IEEE Software 19(3): 43-45.

Brereton, P., Kitchenham, B. A., Budgen, D., Turnover, M. and Khalil, M. 2007. Lessons from applying the systematic litterature review process within the software engineering domain. Journal of Systems and Software 80(4): 571-583.

Brinkkemper, S. 1996. Method engineering: Engineering of information systems development methods and tools. Information and Software Technology 38(4): 275-280.

Braa, K. and Øgrim, L. 1994. Critical View of the ISO Standard for Quality Assurance. Information Systems Journal 5: 253-269.

Buono, A. F. and Poulfelt, F. 2005. Challenges and Issues in Knowledge Management. Information Age Publishing.

CACM 2007. CACM Inside Risks. http://www.csl.sri.com/users/neumann/insiderisks.html. Accessed: 21.06.2007

Carvalho, L., Scott, L. and Jeffery, R. 2005. An exploratory study into the use of qualitative research methods in descriptive process modelling. Information and Software Technology 47(2): 113-127.

Christensen, P. H. 2005. The Wonderful World of Knowledge Management. In Challenges and Issues in Knowledge Management. A. F. Buono and F. Poulfelt (Eds). Information Age Publishing. 337-364.

Collison, C. and Parcell, G. 2001. Learning to Fly: Practical Lessons from one of the World's Leading Knowledge Companies. Capstone Publication.

Conradi, R., Dybå, T., Sjøberg, D. I. K. and Ulsund, T. 2006. Software Process Improvement: Results and Experience from the Field. Springer Verlag.

Conradi, R. and Fuggetta, A. 2002. Improving Software Process Improvement. IEEE Software 19(4): 92-99.

Creswell, J. W. 1994. Research Design, Qualitative and Quantitative Approaches. Sage Publications.

Creswell, J. W. 2003. Research Design, Qualitative, Quantitative, and Mixed Methods Approaches. Sage Publications.

Davenport, T. H. and Prusak, L. 1998. Working Knowledge: How Organizations Manage What They Know. Harvard Business School Press.

Davison, R., Martinsons, M. G. and Kock, N. 2004. Principles of canonical action research. Information Systems Journal 14(1): 65-86.

Deming, E. W. 2000. Out of the Crisis. The MIT Press, Cambridge, Massachusetts.

Denzin, N. K. and Lincoln, Y. S. 1994. Handbook of Qualitative Research. Sage Publications, London, UK.

Desouza, K. C., Dingsøyr, T. and Awazu, Y. 2005. Experiences with Conducting Project Postmortems: Reports versus Stories. Software Process: Improvement and Practice 10(2): 203-215.

Dingsoyr, T., Moe, N. B., Dybå, T. and Conradi, R. 2005. A workshop-oriented approach for defining electronic process guides - A case study. In Software Process Modelling, Kluwer International Series on Software Engieering. S. T. Acuña and N. Juristo (Eds). Boston: Kluwer Academic Publishers. 187-205.

Dingsøyr, T. 2005. Postmortem reviews: purpose and approaches in software engineering. Information and Software Technology 47(5): 293-303.

Dingsøyr, T. and Conradi, R. 2002. A survey of case studies of the use of knowledge management in software engineering. International Journal of Software Engineering and Knowledge Engineering 12(4): 391-414.

Dybå, T. 2001. Enabling Software Process Improvement: An Investigation on the Importance of Organizational Issues. Dr. ing thesis. Norwegian University of Science and Technology.

Dybå, T., Dingsøyr, T. and Hanssen, G. K. 2007. Applying Systematic Reviews to Diverse Study Types: An Experience Report. Proceedings of the ESEM. Madrid, Spain.

Dybå, T., Kampenes, V. B. and Sjøberg, D. I. K. 2006. A systematic Review of Statistical Power in Software Engineering Experiments. Information and Software Technology 48: 745-755.

Dybå, T., Kitchenham, B. A. and Jørgensen, M. 2005. Evidence-Based Software Engineering for Practitioners. IEEE Software 22(1): 58-65.

Earl, M. 2001. Knowledge Management Strategies: Towards a Taxonomy. Journal of Management Information Systems 18(1): 215-233.

Edwards, J. S. 2003. Managing Software Engineers and Their Knowledge. In Managing Software Engineering Knowledge. A. Aurum, R. Jeffrey, C. Wohlin and M. Handzic (Eds). Springer-Verlag. 5-27.

Endres, A. and Rombach, D. 2003. A Handbook of Software and Systems Engineering, Empirical Observations, Laws, and Theories. Addison-Wesley Professional.

Feldmann, R. L. and Althoff, K.-D. 2001. On the Status of Learning Software Organisations in the Year 2001. Learning Software Organizations Workshop. Kaiserslautern, Germany. 2-6

Fenton, N. and Pfleeger, S. L. 1997. Software Metrics: A Rigorous and Practical Approach. International Thomson Computer Press.

Finkelstein, A. and Kramer, J. 2000. Software Engineering: a Road Map. 22nd International Conference on Software Engineering. Limerick Ireland. 3-22

Glass, R. L. 1999. The realities of software technology payoffs. Communications of the ACM 42: 74-79.

Glass, R. L., Ramesh, V. and Iris, V. 2004. An Analysis of Research in Computing Disciplines. Communications of the ACM 47(6): 89-94.

Guzzo, R. A. and Dickson, M. W. 1996. Teams in organizations: Recent research on performance and effectiveness. Annual Review of Psychology 47: 307-338.

Hannay, J. E., Sjøberg, D. I. K. and Dybå, T. 2007. A Systematic Review of Theory Use in Software Engineering Experiments IEEE Transactions on Software Engineering 33(2): 87-107.

Hansen, B., Rose, J. and Tjørnhøj, G. 2004. Prescription, Description, Reflection: the shape of the software process improvement field. International Journal of Information Management 24(6): 457-472.

Hansen, M. T., Nohria, N. and Tierney, T. 1999. What is your strategy for managing knowledge? Harvard Business Review 77(2): 106-116.

Hoyle, D. 2001. ISO 9000 Quality Systems Handbook. Butterworth-Heinemann, London, UK.

Humphrey, W. S. 1989. Managing the Software Process. Addison-Wesley, Reading, MA, USA.

Jacobson, I., Booch, G. and Rumbaugh, J. 1999. The Unified Software Development Process. Addison Wesley Longman.

Kitchenham, B. A. 2004. Procedures for Performing Systematic Reviews. Technical Report TR/SE-0401. Keele University

Kram, K. E. 1985. Mentoring at work: Developmental relationships in organizational life. Glenview, IL: Scott Foresman.

Kram, K. E. and Hall, D. T. 1989. Mentoring as an antidote to stress during corporate trauma. Human Resource Management 28: 493-510.

Kroll, P. and Kruchten, P. 2003. The Rational Unified Process Made Easy - A Practitionare's Guide to the RUP. Addison Wesley.

Kruchten, P. 2001. The Nature of Software: What's So Special about Software Engineering? The Rational Edge. October 2001. http://www.therationaledge.com/

Krutchen, P. 2000. The Rational Unified Process: An Introduction. Addison-Wesley.

Lawler, E. E. and Mohrman, S. A. 1987. Quality Circles - after the Honeymoon. Organizational Dynamics 15(4): 42-54.

Levin, M. and Greenwood, D. J. 1998. Introduction to Action Research -- Social Research for Social Change. Sage.

Lindvall, M. and Rus, I. 2002. Knowledge Management in Software Engineering. IEEE Software 19(3): 26 - 38.

Lindvall, M., Rus, I., Jammalamadaka, R. and Thakker, R. 2001. Software Tools for Knowledge Management. tech. report. DoD Data Analysis Center for Software, Rome, N.Y.

Lyytinen, K. and Robey, D. 1999. Learning failure in information systems development Information Systems Journal 9(2): 85-101.

Mathiassen, L. 2002. Collaborative Practice Research. Information, Technology & People 15(4): 321-345.

Mathiassen, L., Pries-Heje, J. and Ngwenyama, O. 2001. Improving Software Organizations – From principles to Practice. Addison-Wesley.

Moe, N. B. and Dingsøyr, T. 2005. The impact of process workshop involvement on the use of an electronic process guide: a case study. 31st EUROMICRO Conference on Software Engineering and Advanced Applications): 188-195.

Myllyaho, M., Salo, O., Kääriäinen, J. and Koskela, J. 2004. A Review of Small and Large Post-Mortem Analysis Methods. ICSSEA. Paris.

Naur, P. and Randell, B. 1969. Software Engineering: Report on a Conference Sponsored by the NATO science Committee. Garmisch, Germany.

Nonaka, I. and Takeuchi, H. 1995. The Knowledge-Creating Company. Oxford University Press.

Pascale, R. T. 1991. Managing on the Edge: How the smartest companies use conflict to stay ahead. Simon & Schuster, New York.

Paulk, M. C., Curtis, B., Chrissis, M. B. and Weber, C. V. 1993. Capability Maturity Model, Version 1.1. IEEE Software 10: 18-27.

Paulk, M. C., Weber, C. V. and Curtis, B. 1995. The Capability Maturity Model: Guidelines for Improving the Software Process. Addison-Wesley, Reading, MA, USA.

Pedreia, O., Piattini, M., Luaces, M. R. and Brisaboa, N. R. 2007. A Systematic Review of Software Process Tailoring. ACM SIGSOFT Software Engineering Notes 32(3): 1-6.

Pikkarainen, M., Tanner, H., Lehtinen, J., Levonmaa, M., Hyry, H. and Abrahamsson, P. 2005. An Empirical Evaluation of the Process Workshop Approach. 3rd International Conference of Software Development.

Ragins, B. R., Cotton, J. L. and Miller, J. S. 2000. Marginal Mentoring: The Effects of Type of Mentor, Quality of Relationship, and Program Design on Work and Career Attitudes. Academy of Management Journal 43(6): 1177 - 1194.

Riordan, C. M., Vandenberg, R. J. and Richardson, H. A. 2005. Employee Involvement Climate and Organizational Effectiveness. Human Resource Management 44(4): 471-488.

Rising, L. and Derby, E. 2003. Singing the Songs of Project Experience: Patterns and Retrospectives. The Journal of Information Technology Management 16(9): 27-33.

Rus, I., Lindvall, M. and Sinha, S. S. 2001. Knowledge Management in Software Engineering. tech. report. DoD Data Analysis Center for Software, Rome

Scupin, R. 1997. The KJ Method: a technique for analyzing data derived from Japanese ethnology. Human Organization 56: 233-237.

Seaman, C. B. 1999. Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering 25(4): 557-572.

Sjøberg, D. I. K., Anda, B. C. D., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanovic, A. and Vokác, M. 2003. Challenges and Recommendations When Increasing the Realism of Controlled Software Engineering Experiments. In Empirical Methods and Studies in Software Engineering: Experiences from ESERNET. R. Conradi and A. I. Wang (Eds). Springer, Berlin / Heidelberg. 24–38.

SPICE 2007. The Software Process Improvement and Capability dEtermination Website. http://www.sqi.gu.edu.au/SPICE/. Accessed: 20.06.2007

Stata, R. 1996. Organizational learning: The key to management innovation. In How organizations learn. K. Starkey (Eds). Thomson Business Press. 316-334.

Susman, G. and Evered, R. 1978. An assessment of the scientific merits of action research. Administrative Science Quarterly 23(4): 582-603.

ter Hofstede, A. H. M. and Verhoef, T. F. 1997. On the feasibility of situational method engineering. Information Systems Journal 22(6): 401-422.

Vandenberg, R. J., Richardson, H. A. and Eastman, L. J. 1999. The Impact Of High Involvement Processes on Organizational Effectiveness. Group & Organization Management 24(3): 300-339.

Webster's. 1989. Encyclopedic Unabridged Dictionary of the English Language. Gramercy Books.

Wenger, E. 1998. Communities of practice : learning, meaning and identity. Cambridge University Press.

Wenger, E. C., McDermott, R. and Snyder, W. M. 2002. Cultivating Communities of Practice. Boston: Harvard Business School Press.

Whitten, N. 1995. Managing Software Development Projects: Formula for Success. Wiley, New York.

Wickert, A. and Herschel, R. 2001. Knowledge management issues for smaller businesses. Journal of Knowledge Management 5(4): 329-337.

Wiig, K. M. 1997. Knowledge Management: An Introduction and Perspective. Journal of Knowledge Management 1(1): 6-14.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. and Wesslén, A. 2000. Experimentation in Software Engineering. Kluwer Academic Publishers.

Yin, R. K. 2003. Case Study Research, Design and Methods. Sage Publications.

Zelkowitz, M. V. and Wallace, D. R. 1998. Experimental models for validating technology. IEEE Computer 31(5): 23-31.

Aaen, I., Arent, J., Mathiassen, L. and Ngwenyama, O. 2001. A Conceptual MAP of Software Process Improvement. Scandinavian Journal of Information Systems 13: 81-101.

Appendix A: Selected papers

In this appendix we have included the seven papers that have contributed the most towards the work presented in this thesis. We present them here in chronological order. The papers are:

- P1: Harvesting Knowledge through a Method Framework in an Electronic Process Guide
- P2: Tailoring RUP to a defined project type: A case study
- P3: A study of a Mentoring Program for Knowledge Transfer in a Small Software Company
- P4: Defining Software Processes Through Process Workshops: A Multicase Study
- P5: Tailoring and introduction of the Rational Unified Process
- P6: Improving the Effectiveness of Root Cause Analysis in a Retrospective Method: a Controlled Experiment
- P7: Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts and Research Methods Used

The papers have been uniformly formated for presentation in this thesis.

P1: Harvesting Knowledge through a Method Framework in an Electronic Process Guide

First published in Proceedings of the 7th International Workshop on Learning Software Organizations (LSO), Kaiserslautern, Germany, 2005, 107-111
Also available from K.-D. Althoff et al. (Eds): WM 2005, LNAI 3782, pp. 86-90, 2005.

Finn Olav Bjørnson, Tor Stålhane
Department of Computer and Information Science,
Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
{bjornson, stalhane}@idi.ntnu.no

Abstract. A key leverage for small software consultancy companies is the collective knowledge possessed by their consultants. There have been some studies in the literature on how to harvest and transfer this knowledge, but most studies are aimed at large multinational corporations. In this paper we describe an ongoing research project, aimed at improving knowledge sharing in a small software consultancy company through the use of a method framework in an electronic process guide coupled with an experience repository.

1 Introduction

Small software consultancy companies have to leverage their position in the market to stay ahead of their competitors. One way to achieve this is by providing their customers with tailored solutions to their problems. They can do this by drawing upon the collective knowledge of their consultants. When the company is small and the consultants are spread over the sites of many customers, it becomes difficult to gain access to and draw upon the collective experience of all the co-workers. Consequently the solutions provided by the consultants might not be of sufficient quality to make their customers return to the company when they need consultants for a new project.

One solution to the problem with a dispersed workforce is experience repositories. A lot of research has gone into this field, however most of this research has been focused on large companies and little data exists on the application of this in small companies [1, 2].

In [3] the authors examine challenges facing small businesses when implementing knowledge management efforts. Small businesses are particularly vulnerable to knowledge erosion, yet they seldom have the time and resources needed to implement the knowledge management programs described for larger companies. However, the authors suggest that small businesses can benefit just as much from well thought out knowledge management efforts.

According to [1], which describes the successful use of an experience repository in a small software company, detailed data on its use and structure can be used to better understand how experience supports activities in the company. This can in turn lead to improvements in experience management concepts, techniques and tools.

In this research report we describe our work in a small software consultancy company that wish to manage their knowledge through a method framework implemented in an Electronic Process Guide (EPG) coupled with an Experience Repository (ER).

2 Context

The company we investigated currently has 17 employees. Their main activities are hiring out consultants as developers, developing complete solutions for customers and hiring out consultants as advisors for selecting technology, strategy or process. Typically, no more than four to five consultants are at any time working for the same customer.

The managers of the company wish to leverage the company in the market by providing solutions to the problems of their customers. The solutions should make them stand out and increase the probability that the customers later returns with new projects. In order to do this, they wish to foster an environment were all ideas and knowledge are shared freely among the employees, and where the employees can draw upon the experience of each other to provide good services to their customers. This work is difficult since a lot of the employees at any given time are out at the site of customers where they don’t have direct access to their colleagues.

To remedy this situation they wish to collect the experience of their employees in an Experience Repository (ER). This will allow their employees to have easy access to the experience of their coworkers.

3 Method

Due to the cooperative nature of this research project, we decided to adopt action research as our approach. The most prevalent description of action research is found in [4]. The approach requires the establishment of a client-system infrastructure or research environment. In our case this was already taken care of through the researchers’ and company's involvement in a mutual research program. The approach further specifies five identifiable phases, which are iterated: diagnosing, action planning, action taking, evaluating and specifying learning. This paper sums up our work and findings from the initial phases and what effect this has had on the development of the new tool. The plans for the next phases are outlined up in section 6: Future work.

For the initial diagnosing phase, we decided to use semistructured interviews. We scheduled interviews with 12 of the employees. The interviews were carried out using an interview guide. Basically we wanted answers to three questions: What was the current approach to knowledge sharing, what should the new tool contain, and what kind of functionality should it provide? All of the interviews were taped using a

dictaphone and were subsequently transcribed. The material was then coded and analyzed using the constant comparison method and the NVivo tool [5].
The problem with the adopted approach is that our results will be difficult to generalize due to our single case. Rather they will contribute to the understanding of the concepts of Experience Repositories. If the results from our study should coincide with the research literature some generalization might be possible.

4 Interview Results

The company seemed to have a good environment for informal sharing of experience in that people knew one another and knew whom to contact if they were stuck. There did not seem to be much formal gathering of experience. If experience from a project was collected, it was mostly done in an ad hoc manner, and it was not easily available. The gathering of experiences today was mostly done through private initiative and saved for personal use.

Lately a few employees had begun using post mortem analysis [6] at the end of their projects, but they did not have a place to structure and access this information. The fact that a lot of work was done at the site of customers was also seen as a hindrance to collecting project experience. It seemed to be easier to get help with technical problems than problems related to process. More structure and information related to process was seen as desirable.

When asked about what information they wanted the new tool to contain, the employees provided us with a myriad of elements. A few, however, was mentioned more often than the others: document templates, patterns, a good process, help with customer relations and practical experience.

Document templates were seen as potential help to increase productivity. Both inexperienced and experienced project managers saw a benefit from having a set of standardized templates in order to save time on documentation.

Patterns were also mentioned as something that should be readily available. Good ideas and smart solutions that other people had thought of were worth repeating. However, the employees stressed the need for trust. It was important for them to know that a pattern could actually deliver what it promised.

A good development process and the need for help with questions related to process was often mentioned during the interviews. This need was considered especially important for the start-up of new projects. Inexperienced project managers expressed a need for a process that would help and guide them through the initial phases. Experienced managers expressed the need for a process that would help them keep on track throughout the project. A well-defined process was also seen as something they could market to their customers to gain an edge over their competitors.

The employees often mentioned the need for guidelines and advice on how to improve customer relations. There was a broad agreement that more customer involvement would enhance the quality of the end product. The employees agreed that there had not been a lot of focus on this in the past and that guidelines for this would be most welcome in the new method framework.

When it came to choosing a process, a template or a pattern, the employees would like to know what kind of experience others had made when using these items. They saw a great potential in linking the experience of the company's developers to templates, patterns and processes, in order to be able to assess them for their own projects based on their colleagues' experience.

5 Initial work and Challenges ahead

After the initial interviews we moved on to the action-planning phase of our research. This phase consisted of meetings with the company where we presented the result of our interviews. The interviews indicated that there was a demand for a tool that would help the employees to share and structure their experience, especially experience surrounding the development process. It also indicated that the culture of the company supported free sharing of information and experience, and that the employees saw the benefits of using such a tool as the management was suggesting.

With the support from the employees established, we arranged a discussion on the functionality and the content of the new tool. It was decided that the company should develop an empty method framework tailored to the development process of the company. This framework would be implemented in a dynamic EPG, which would then be coupled to an ER. The employees would use this tool to enter their experience related to roles, artifacts and activities. The goal is to create a process guide based on the collective experience of all the employees in the company, which can then be used to increase the quality and consistency of their work. Both the decision to couple the ER to the process of an EPG and making the tool highly interactive to enable fast feedback is supported by [7] which describes good practices regarding ER and [2] which describes a successful implementation of an EPG/ER

After the meeting where this was discussed, we moved on to the action-taking part of our research. The company put one consultant on the project of working out a method framework. The framework was based on the Rational Unified Process (RUP), and was tailored to the company's process. During this process the input of both employees and scientists was sought in order to make the framework as similar to the current practice as possible.

One of our main challenges in the time ahead will be to keep the ER alive. An ER that is not used by the developers is of no value to the company. Experience from other ER initiatives [8, 9] has shown that there are three factors that influence the use of an ER:

- The ER must contain a minimum amount of experiences that can be searched. The amount of experience available is critical. If there is little experience available in the ER, the developers will neither use it nor contribute their own experiences to it.
- The experience that is found must be considered to be relevant for the developers in their day-to-day work. It must help them to do a better job and it

must be up to date. One of the most de-motivating things that can occur when using an ER is to find an experience with and interesting title but with outdated contents.
- It must be possible to establish a community of practice [10] based on the ER contents. This means that not only must the experience be relevant – it must be possible to discuss, and augment existing experiences, that is; the ER must work as a forum where people can exchange ideas.

All of these mechanisms are used to keep up the interest for the ER among the developers. On the other hand, the interest can only be kept if the content is good. In order to meet these challenges we will use several strategies. The most important mechanisms to achieve our goals are to:

- Keep the ER open. As a consequence of this, everybody can add his or her own experiences. There will only be one restriction – all input must be traceable to the person that contributed it.
- Build discussion treads. These are important both to keep the experiences up-to-date and to keep the community of practice alive.

6 Future Work

When the framework is finished and implemented in the EPG/ER tool it will be presented to the employees. After this, the employees will enter into a period of filling up the framework with relevant experience. The next challenge for the scientists will be to come up with good methods for extracting most of the experience of the employees in a way that is not too intrusive to the regular work of the company, yet still captures the most crucial knowledge.

After an initial trial period the tool will be approved for use in projects. The role of the scientists then switches to an observational role. We plan on following the use of the EPG/ER for two years (the remaining period of our research project). By collecting information along the way and comparing it with the research literature, we hope to be able to ascertain how successful the knowledge initiative have been for the company and how it might apply to companies in similar contexts.

References

[1] Louise Scott, Ross Jeffery: *The Anatomy of an Experience Repository*, Proc. International Symposium on Empirical Software Engineering, 2003

[2] Felicia Kurniawati, Ross Jeffery: *The Long-term Effects of an EPG/ER in a Small Software Organisation*, Proc. Australian Software Engineering Conference, 2004

[3] Wickert Anja, Richard Herschel: *Knowledge management issues for smaller businesses*, Journal of Knowledge Management, vol 5, no. 4, pp. 329-337, 2001

[4] Susman G., Evered R.: *An assessment of the scientific merits of action research*, Administrative Science Quarterly, 23(4), pp. 582 – 603, 1978

[5] Web: http://www.qsrinternational.com, last visited 06.09.04

[6] Birk Andreas, Dingsøyr Torgeir, Stålhane Tor: *Postmortem: Never Leave a Project Without It*, IEEE Software, vol 19, no 3, pp. 43-45, 2002

[7] Kurt Schneider, Jan-Peter von Hunnius: *Effective Experience Repositories for Software Engineering*, Proc. 25th International Conference on Software Engineering, 2003

[8] Louise Scott, Tor Stålhane: *Experience Repositories and the Post Mortem*, Proc. Learning Software Organisations, 2003

[9] Hauge Tor-Erik, *Reuse in IT-companies, evolution and trends*, Master Thesis University of Stavanger, 2003, (in norwegian)

[10] Wenger Etienne: *Communities of Practice*, Cambridge University Press, 1998

P2: Tailoring RUP to a defined project type: A case study

Presented at the 6th International Conference on Product Focused Software Process Improvement, Oulo, Finland, 2005
Available in F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2005, LNCS 3547, pp. 314-327, 2005. © Springer-Verlag Berlin Heidelberg 2005. Reprinted with kind permission of Springer Science and Business Media.

Geir K. Hanssen[1], Hans Westerheim[1], Finn Olav Bjørnson[2]
[1] SINTEF ICT, N-7465 Trondheim, Norway
{geir.k.hanssen, hans.westerheim}@sintef.no
[2]NTNU, N-7491 Trondheim, Norway
finn.olav.bjornson@idi.ntnu.no

Abstract. The Unified Process is a widely used process framework for software development. The framework is covering many of the roles, activities and artifacts needed in a software development project. However, a tailoring of the framework is necessary to fit specific needs. This tailoring may be accomplished in various ways. In this paper we describe a concrete attempt to tailor the Rational Unified Process to a defined project type; a Mainstream Software Development Project Type. The paper has focus on the process of creating the tailored Rational Unified Process as well as the resulting Rational Unified Process. The paper makes some conclusions and has a proposition for further research.

1 Introduction

The Unified Process [1] and the commercial variant, the Rational Unified Process, RUP [2] are comprehensive process frameworks for software development projects. RUP defines a software development project as a set of disciplines, e.g. requirements handling, implementation etc., running from start to end trough a set of project phases. A project is performed by a group of actors, each having one or more well defined roles. Each role participates in one or more activities producing one or more artifacts. A discipline can run in iterations, that is, repetitions within a phase. Activities, roles and artifacts are the basic process elements of RUP.

However, RUP is a comprehensive framework, meaning that it is a more or less complete set of process elements that has to be tailored to each case as no project needs the complete set of elements.

Jacobson, Booch and Rumbaugh says in [1] p.416: *"It [RUP] is a framework. It has to be tailored to a number of variables: the size of the system in work, the domain in which that system is to function, the complexity of the system and the experience, skill or process level of the project organization and its people."* Further on they say: *"Actually, to apply it, you need considerable further information."*

So, it is clear that RUP needs to be tailored, downscaled and specialized to the context of use. Looking at literature there are not many guidelines on doing this [3], [4], [5] although the need for good practical guidelines and advice definitively is present.

While discussing adaptation of RUP, it is important to have in mind that RUP is a methodology suited for some software development projects, not all. Before you consider using RUP as a basis for your processes you should think of what you really need and what you really do not need. RUP is designed to support four basic properties of software projects: use-case based customer dialogue and documentation, an architecture focus, iterative processes and incremental product development. The idea of adapting RUP is to make it fit each specific project not loosing these properties. It is important to keep the integrity of RUP as a framework. So, an adapted or downscaled variant still defines a project in terms of phases and still describes the work as a complimentary set of disciplines. However, some disciplines may be omitted or even added.

The goal of this paper is to provide others considering remodeling and adapting a process framework in general, and RUP particularly, an insight in how this has been done in a small software company. Some aspects of the specialization process seems to have been working well, others not. This paper presents the adaptation process and also gives an analysis of this process and its result.

The work detailed in this article was carried out as part of a national research project in process improvement and software quality called SPIKE. SPIKE is short for Software Process Improvement through Knowledge and Experience. The participants are SINTEF, NTNU, the University of Oslo and several partners (companies) in the Norwegian ICT-industry. The industrial partners are interested in improving their development process, and are seeking concrete processes and methods to help them deliver high quality software with shorter time to market.

The paper starts with a **Theoretical context**, giving a brief introduction to methodologies and frameworks and various strategies of making these fit specific project needs of process support. It then describes the action research as the **Research method** of choice. The rest of the paper is arranged according to the research method phases; **Diagnosing**, **Action planning**, **Action taking**, **Evaluating** and **Learning.** Finally a **Conclusion** is given and **Further research** suggested.

2 Theoretical context

2.1 Software Development Methodology and Frameworks

The term methodology is defined as "A body of methods, rules, and postulates employed by a discipline: a particular procedure or set of procedures" by the Merriam-Webster dictionary [6]. Basically, a methodology describes how someone, e.g. an organization performs a task, e.g. software development. In a broad sense, a software development methodology describes aspects such as how to communicate with customers, sales strategy, how to describe requirements, use of tools, test practices, documentation, planning, reporting and so on. In our context we talk about methodologies for running projects with a defined customer having more or less defined goals initially. Besides describing techniques, roles etc. most methodologies are based on a set of basic values. Examples are *User centric, Architecture centric, Agile, Risk driven* and many more. RUP has four basic values: *Use-Case Driven, Architecture-Centric, Iterative* and *Incremental.* These values should be retained regardless of how RUP as a framework is adapted. A methodology framework is a comprehensive description of a methodology describing approximately all possible details of almost all possible processes within the scope of the framework. This means that a framework is not a description of a specific case; it is a foundation for adaptation. The challenge is how to adapt it to each case (project) and keep the basic values and features of the framework.

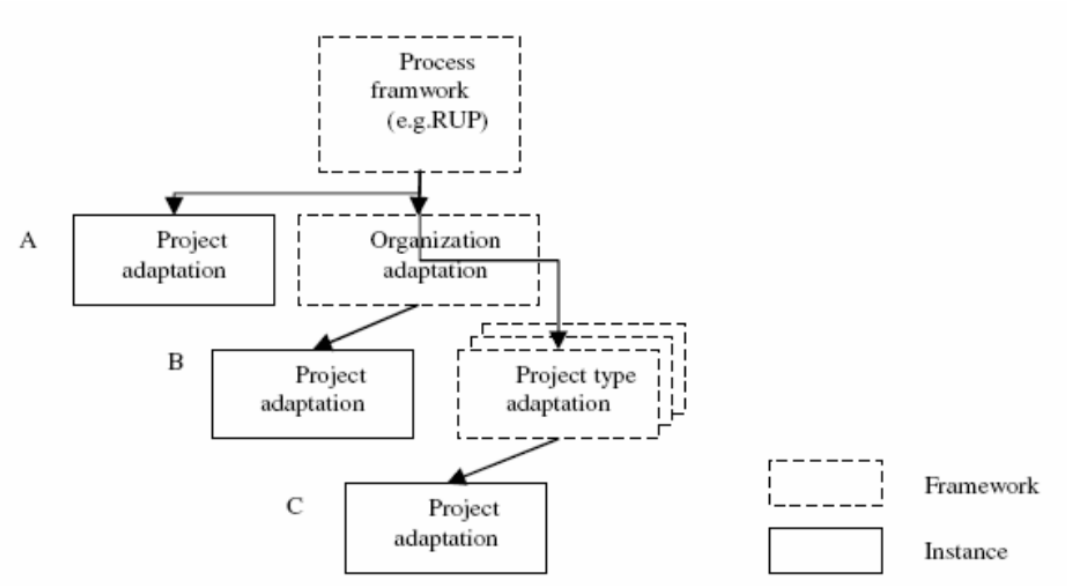

Figure 1 Three possible approaches for adaptation

2.2 Adaptation of RUP

The process of adapting RUP can possibly take many forms. IBM Rational, the provider of RUP has defined the *Process Engineering Process (PEP*) [5]. This is a comprehensive adaptation process requiring a fairly big amount of resources (people and time). This may very well be appropriate for larger companies, but for the small ones this process may be too expensive.

Adaptation of a framework, such as RUP, can take one of (at least) three approaches; see Fig. 1. The starting point is a process framework that is general and complete with respect to tasks, roles and products. In approach A, the framework is adapted, in one step, for each project, thus representing a heavy job in each case. This can be justified for large projects where the initial adaptation process itself becomes only a small part of the total amount of work being done in the project. In approach B, the organization does an up-front adaptation producing a subset of the framework, still being a framework, but now tuned to the organizations general characteristics. This is the intentional process of

PEP. In approach C, the organization first identifies and describes a set of recurring project *types*. Having knowledge of characteristics and differences of these types, an adaptation is done for each type.

No matter which approach being used; in the last step, a final adaptation is done to each case (project). The agility of this final fine tuning increases with respect to the extent of the up-front adaptation.

This is a general view of methodological adaptation or down-scaling. It applies to many types of process frameworks, including RUP. Further on, adapting RUP in practice means to decide on which process elements to keep, remove, alter or add. These decisions can be based on assumptions, experience, goals and visions. It is the quality of this underlying knowledge and experience that determines how good these decisions are.

Running an adaptation process, in general, can be seen as a knowledge management activity as experience and knowledge, both tacit and explicit, is being structured, documented and communicated trough the resulting software process description [7].

3 Research Method

Due to the cooperative nature of this research project with company external researchers acting partly as consultants and partly as researchers, we decided to adopt action research as our approach. Avison et.al. [8] describes action research as: “unique in the way it associates research and practice. … Action research combines theory and practice through change and reflection in an immediate problematic situation within a mutually acceptable ethical framework."

Susman and Evered [9] described an approach to action research that is widely used today. We have adopted elements from this approach in our research project. The approach requires the establishment of a client-system infrastructure or research environment. In our case this was already taken care of through the researchers and company's involvement in the SPIKE research program. The approach further specifies five identifiable phases, which are iterated: *diagnosing*, *action planning*, *action taking*, *evaluating* and *specifying learning*. This report details some of our findings and experiences from the initial phases. Our coverage of the evaluating and learning phases are based on our own observations of the process so far. A more thorough evaluation will be carried out as the company takes the resulting process description into use in real projects.

In the diagnosing phase, we used semi structured interviews and workshops with key employees. We interviewed five employees concerning their general experience with projects in the company. This gave us the material to do a more focused interview with five other employees concerning their specific experience with RUP in the company. In addition to this, several work-meetings were held with the management of the company where the SPI approach was discussed.

In the action planning phase, the researchers made a literature survey of the field of adapting RUP. It was decided to identify possible project types run by the company. This was done during two iterations, the first one a bottom-up approach, the second one a top-down approach. The top-down approach led to definition of three project types. In order to adapt the first project type, it was decided that the researchers should facilitate a workshop where key employees were invited to define the adapted process.

The workshop was carried out as part of the action taking phase. It was carried out over two days, since it was discovered that we needed more time than originally planned. At the first day we noted that the lack of a RUP mentor slowed the process considerably due to a lot of discussion on what was actually meant by the different concepts. At the second day, one such mentor was present, and the process was much more fluent. The result from the workshop was a coarse RUP skeleton, which was given to the company for more refinement. The company has conducted two internal workshops with its employees to refine the process. In addition they have initiated a project to put this information on a Wiki web, in order to make the adapted process available to all employees.

As the project moves into the evaluation phase, the role of the scientists switches to a more observational role. We plan on following the use of the adapted process for several development projects. By taking measures along the way we hope to be able to ascertain how successful the initiative has been for the company in its current context.

4 Research Context

The company described in this case is today a Norwegian software consultancy company with 50 employees, located in two different geographic offices. During the work described in this paper the company was declared bankrupt, and then restarted with new owners. The first part of the action planning and action taking described in this paper took place before the bankruptcy. The first attempt to identify project types was done, using a bottom-up approach. Just before the bankruptcy this approach was evaluated and the company and the researchers decided that this approach did not work. The company then had about 70 employees.

When the company was restarted, the researchers continued to the work mainly together with the other office, but the focus was still the same, and the most actual people from the company did not change. The company is mainly developing software systems with heavy back-end logic and often with a web front-end, typically portals. However, they also develop lighter solutions with most emphasis on the front-end.

The company acts as an independent software supplier, though there are close relationships to the biggest customers. Of the 50 employees today, 35 are working as software developers. Java and J2EE are used as development platform. The domain of which the company develops software is mainly for the banking and finance sector, as well as for public sector. The company has run 50 development projects within the bank and finance sector the last twelve years, and about 30-40 projects within the public sector the last 15 years.

Four employees are certified RUP-mentors acting as advisors in other SW-organizations, in addition to this they run training courses in RUP and related subjects. The company utilizes their high competence in RUP and most projects are more or less inspired by RUP, however, the company's management has seen a need and a possibility to improve their use of RUP.

5 Diagnosing

The decision to initiate a project-type specific adaptation process was made by the company when SPIKE started.

The diagnosing phase was initiated by a few workshops where an internal software development process group defined the strategy in cooperation with the authors. With the past experience in mind they decided to go for a top-down approach, starting out with the complete RUP set of process elements and then customize this set to a set of defined project types. This decision was supported by the findings in two rounds of interviews in the company.

This phase of the work was conducted mainly by three different motivations:

1. The researchers needed more insight into the company, the development organization of the company, as well as the most recent software development projects conducted by the company.
2. The company needed to be more conscious about its own use of RUP; these interviews were means in that respect.
3. The use of RUP in the company needed to be documented as a basis for further work; this includes the overall use, but also strengths and weaknesses by the use, in the view of people working in projects in the company.

Interview 1: General experiences from project work

5 employees having various project experiences were interviewed. The roles of these persons were developer/systems architect, project leader/manager, project leader, senior developer and developer/architect/DBA.

The intention of this group of interviews was to get a perception of common problems and challenges in development projects to establish a basis for process improvement initiatives in the company.

The interviews revealed that the customer dialogue could be better (requirements handling and project planning). The reuse of templates could be better. It is too much documentation formalism. Estimates often fail and there is a need of better change management

Interview 2: Special experiences with RUP

Another group of 5 employees was interviewed to get a view of their experience using RUP. The role of these persons was developer, developer/project leader, developer/project leader/test leader, project leader/requirements responsible, and customer contact.

All of the five had some knowledge and experience with RUP, some had participated on internal courses, and some had read literature. However, none had thorough knowledge and experience. About the practical use, it seemed that RUP was used just to a small extent, it depended on the type of project. The reason for this may be superficial knowledge of RUP and that some felt that RUP does not fit their needs.

These two iterations of interviews gave no clear answer, however they indicate that RUP and the use of it can be improved. The summary from the interviews was used to decide to initiate an adaptation process as described in this paper.

6 Action planning

Projects conducted by the company varied with respect to domain, degree of experimentation, technology, contract form etc. In addition, most projects were too small to initiate a project-specific specialization (ref Figure 1, approach A). However, it seemed that this company usually ran a few similar types of projects. This lead to the idea to define a set of processes fitting each type of project. The idea is that this will reduce the need of a costly up-front specialization per project and also avoid an expensive per-project adaptation. Based on this realization the company decided to try out approach C in figure 1 in cooperation with the authors. The company would define a set of project types which covered most of their projects and define a downscaled RUP to each project type.

To define a set of project types we decided to hold a workshop to identify the company’s three main project types based on a top down approach. The reason for selecting the top down approach was the company's previous failure to define project types based on a bottom up approach. The participants of the workshop consisted of people from the company with a complimentary and thorough knowledge of the company's software development projects, some of them were also RUP mentors. It was also decided that the participants should come up with a classification system to describe and distinguish the three project types.

Given the three distinct project types, the challenge was how to adapt RUP to each project type. There seemed to be wide agreement that adapting RUP was necessary, yet little information was available on how to actually carry out this adaptation process. What little information was available consisted of rather complex and expensive methods. Instead of using any of these methods we decided to go for a simpler and pragmatic approach. It was decided that the researchers should facilitate a workshop where key employees were invited to define the adapted process. The structure of the workshop was planned by the researchers based on their experience and input from the literature, and the participants were selected by the company based on their experience with different disciplines.

After this workshop the material was left to the company to refine and document with little input from the researchers.

7 Action taking

The RUP adaptation itself was separated in four main phases:
A. Defining the project types
B. The definition of the mainstream project type
C. Maturing the downsized RUP
D. The initial documentation of the mainstream project activities

A: Defining the project types

We conducted a workshop where five participants from the company, representing a group with a complimentary and thorough knowledge of software development projects in general and RUP in special (some of them RUP mentors), were allowed to define three to four common types of projects. To be able to distinguish and describe the project types we defined a simple classification system. During a series of workshops a group representing all project roles identified a set of project capabilities to be used to describe the project types. A project capability, in this context, is a feature or a characteristic that is general to all projects but where the size or weight does vary. We identified 13 characteristics; business critically for the customer, technology knowledge, access to resources, risk, test environment, size, degree of reuse, contract form, project team, exposure, customer orientation, system integration and scope.

The three selected types of projects were Mainstream Projects, Push-button Projects and Greenfield Projects. Here presented with a few characteristics:

Mainstream projects	**Push-button projects**	**Greenfield projects**
- integration with other systems are important - the technology are well known - the size are initially unclear - the risk is moderate	- the technology is well known - low-risk project - well defined project size - often a fixed price project	- need of extensive research and innovation - the size are initially unclear - high risk project - newer fixed price

B: The definition of the mainstream project type

We selected the mainstream project type since this was the most important type for the company with respect to earning. The two other project types will be handled later.

Originally we envisaged a workshop to define a list of RUP elements necessary for the different disciplines and phases. The result from this would be a list that needed some refinement and quality assurance before it could be documented and put into use in a project. The method we ended up with was not far from this. It consisted of two days where the focus was defined by RUP elements viewed from the point of view of either the RUP phases or the RUP disciplines.

On the first day we gathered a group of employees with relevant experience from mainstream projects, meaning people that have both the theoretical and practical knowledge of RUP from projects as well as experience relevant to the defined project type. We tried to ensure that all the disciplines of RUP should be covered by the experience of the workshop participants. The process of the initial workshop was as follow:

1) The workshop facilitators (the researchers) explained the defined project type for the group and this was discussed. This was done to establish a common mindset for the rest of the work.
2) We used a whiteboard with a vertical lane for each RUP-phase (inception – elaboration – implementation – transition) to document opinions of what was especially important for each phase (based on practical experience). The workshop facilitators asked questions such as: *What is usually a challenge in this type of project? What type of methodology support do you need? What has used to work well?* All this to sharpen the focus of what is important for the project type and how a defined process can support it.
3) The workshop facilitators displayed a list of all RUP process elements using a video projector. A process element was a defined role, artifact or activity. The elements were ordered per RUP discipline. Starting at the top the group made decisions for each element whether to keep, remove or alter the element. The two previous steps was used as basis for taking decisions and was referred to during the selection process. However, this turned out to be a circumstantial process. The group and the workshop leaders agreed to only focus on *artifacts*, thus speeding up the process to a practical level. When an artifact was removed, this implicitly also indicated how roles and activities should be affected. An example of a artifact that was decided to be deselected is 'Capsule'. The RUP documentation explains that this is an artifact *"Used only for the design of real-time or reactive systems.."*, thus not relevant for the Mainstream project type described and discussed in step 1.

Step 3 was not finished by the end of the first day. One of the main reasons for this was that there was no RUP mentor present. Subsequently there was a lot of argument over what the different RUP concepts actually meant, and a lot of the time was spent searching for information. Another reason was that we initially tried to define artifacts, roles and activities; this took up a lot of time, thus it was decided to just focus on artifacts. Since the list was not finished at the end of the day, it was decided to spend a second day to finish the work. In the second day we only focused on artifacts and the company provided us with a RUP mentor. This time the process worked more fluently and we were able to finish the list of adapted RUP elements to mainstream projects.

C: Maturing the downsized RUP

Due to the composition of the members of the workshop, some disciplines were better covered than others. This sparked some discussion in the company on how to proceed. They found it necessary to involve more people to increase the information on certain disciplines, and it was decided that to increase the usefulness of the process it was necessary to run more iterations to gather experience from all the disciplines.

Having compiled the list of process elements the company continued the process by involving more of the employees. This to incorporate more relevant experiences and, not at least, to establish a common ownership. The focus turned from selecting/deselecting process elements at a very low level to focusing on best practices, in this case meaning to focus on vital project activities. Their next step was to define critical activities for each phase of RUP. This was done in a separate internal workshop. For each phase they held a discussion on what the critical activities were. When they agreed on an activity they found a descriptive name for it and proceeded to answer two questions: 1) What is accomplished by performing this activity? And 2) What is the risk of not performing this activity, or not performing it properly?

The name of the activity and the answer to the two questions was written on a piece of paper and post-it notes and put on a large paper that covered the wall. There was one such paper for each phase.

D: The initial documentation of the mainstream project activities

Having specialized RUP, or any other process for that matter, does not complete the job. The result must be brought out to the frontline people – the project leaders, the developers, the architects and so on. They must have the information at their fingertips in the actual situation of use in a form that makes them want to use it. There is a variety of practical ways of communication this information, from simple documents, to simple web-pages, to comprehensive hypertext documentation. Rational offers an electronic process guide that documents RUP in detail (RUP Online). This is a knowledge base with a web interface that describes roles, activities and artifacts (and templates for these – all arranged within the phases and disciplines of RUP. However, RUP online is comprehensive and may be more confusing than helpful to project members in need of specific project support. Any documentation of the process must reflect the modifications resulting from the specialization process.

Instead of using the tools from Rational, the company decided to establish a simple Wiki-web [10] with just-enough information and functionality to get the message out. This web does not resemble to the RUP-online documentation which holds a well of details. This Wiki can be seen as a common electronic whiteboard, where all users have more or less full access to the information and the rights to update it.. This Wiki Web is a company internal web-site that in simple terms describes the outcome of the workshops and the company internal process work. It explains the characteristics of the project type(s) so that the user can evaluate how well the variant suits the actual project and can also be used as a checklist to plan the project. The simple process documentation on the Wiki Web references RUP Online (web link) to lead the user to helpful descriptions and templates. A Wiki-Web also allows the users to add information thus being a dynamic process repository. One idea (not yet tested) is to store project experiences together with the process descriptions to offer later projects an insight into specific and relevant experience.

The resulting process description

The resulting process documentation, presented trough the Wiki-web, is much simpler than we initially would think. It is more a guide into RUP than an independent complete process guide.

The process definition of the Mainstream type of projects is simply a list of critical activities where each activity is defined by 1) a title stating the purpose of the activity, 2) a short description, 3) the context of the activity, 4) reasons for why this is an important activity for this project type, 5) risks by omitting the activity, 6) a checklist for completion of the activity and 7) recommended problem solving approach. All these seven parts are presented on one page.

These activities are arranged with respect to the standard phases of RUP and also has some links to relevant information in RUP Online, e.g. to templates etc. This simple description is intentionally on a high level, omitting most of the details of RUP. The Wiki-web offers this information to all project members via the intranet. A separate area is created for each project where the project members document their best practices, templates used, comments to the process. In general, this is an experience reporting tool that communicates practical experiences for a given project type to others.

The case company has constituted a process group that continuously updates and refines the content of the Wiki based on real experiences being reported on the Wiki.

8 Evaluating

The company did from the beginning focus on project types. During the work described here, two different approaches were tried in order to define different types of projects. The bottom-up approach was tried first, and then the top-down approach. The bottom-up approach did not succeed as it became too complex to document a big amount of project experiences and identify a few common variants of RUP. During the workshops where this approach was tried, it was clear that the participants felt that the project types in some ways were defined already, but not given. The company had an informal definition of project types, not named ones, but with some consensus among the developers what these types were. In the workshops we tried to keep the entire focus on the characteristics of the project types, and the participants were not "allowed" to state types of projects. This approach clearly made the participants frustrated, and the approach did not bring up any defined project types based on the defined characteristics.

We did succeed with a top-down approach to defining a set of project types – starting by loosely naming typical types and then describe typical aspects trough a workshop. The participants were told to name three project types in the beginning, and this strict introduction seems to have helped the participants to reflect over what is really separating the different types of projects there were working on. The three types were relatively easy to identify and name. During the work these initial types were kept, and the belief that these were the important types grew. Even though the initial try with focus on project characteristics did not succeed, this attempt kept the focus on project characteristics during the whole work described here, and the participants were more conscious about what is a project type than the case might have been without the first try. The researchers therefore would like to recommend trying to keep focus on different aspects and characteristics of software projects.

During the work the focus has been on one type of projects only. The company did pick the type of project which was most important with respect to earnings and risk control, and the first attempt to tailor RUP was for this single type only. This focus seems to have been an important factor when it comes to the ability to tailor RUP. Having a common, well defined, mindset makes the decisions easier and the result simpler and more focused.

In this case study, a discussion of which tool to use for the documentation and deployment of the tailored RUP was postponed to a moment when the discussion about the content of the tailored RUP was in place. Adapting and documenting RUP or any other methodological framework is not done solely using a tool. The most crucial part of such a job is to involve a broad group of people having through experience with both the framework and – not at least – practical project work. The work in this case supports this presumption.

Employees in this company have knowledge of RUP above the average of what we have seen in analogous software development organizations in Norway. The work in the company shows that it is important to have a tailoring process that must be based on experience; it can be seen as a knowledge management, and documentation, process. Despite the company's knowledge of RUP, running such a process has not been easy and straight forward at all. The strategy has changed during the course of work based on new insights and achieved results (or lack of such).

9 Learning

Our motivation intentionally was to work together with the case company to adapt the RUP. We decided to try to keep it as simple and inexpensive as possible. The two authors that participated actively in the start worked with a small group from the company, thus reducing the total time spent. We also tried to use RUP as a heavy foundation by accepting the general characteristics of the method, such as the phases and the disciplines and go straight to the low-level details; the process elements. But this did not seem to be the best way. The process did become simpler and simpler as the work progressed. This helped the involved people keeping focus on what’s most important; what type of process support is really needed in the projects based on experience. When starting out we intentionally did not take a standpoint with respect to *how* to document and disseminate the resulting process description. We looked into the suite of tools offered by Rational, but regardless of the rich features in those tools the company ended up with a very simple form of tool support for documentation and communication of the result, the Wiki web. In general it seems that the adaptation is best done as a simple, pragmatic process not as a heavily up-front planned and strictly managed process. It seems that the good old KISS-strategy once again have proven its superiority; Keep It Simple Stupid.

Some specific experiences from the tailoring workshops

Having good knowledge and experience is important to ensure sound decisions on how to adapt RUP. This however presupposes that such experience is available within the organization, which was the case in the project that this paper is based on. If the overall knowledge of RUP is weak the group can be strengthened by hiring a RUP-mentor. The mentor is a certified expert that will be in position to answer questions and explain details of RUP.

Having a group working through the three steps of the initial workshop should take about one working day, given that the workshop leaders have prepared the work, the focus is on artifacts from a discipline point of view, and that there is a RUP mentor present to explain any uncertainties. To ensure a good result it is vital to include people with experience from all the disciplines of RUP.

Do not try to gather too much information in one single workshop. Concentrate on one issue at a time.

It is important to be patient; the outcome of the initial workshops was nothing but an altered list of RUP process elements. This list has to be matured and quality assured before it can be documented and put into use in projects.

10 Conclusion

We have presented a simple pragmatic method for adapting the RUP to a specific project type in a company. The method involves a series of workshops in which the key success factor seems to have been focus. Focus both through a specific project type, specific process elements and through phases or disciplines. Another key success factor is that a workshop consists of persons with the proper experience with regards to the focus.

The focus on a specific project type seems to have kept the participants on track throughout the adaptation process. It seems to have eased the process since everyone had a clear concept of what should be done in that particular project type. However, the benefits from making a project type adaptation as compared to making a project- or a company specific adaptation have yet to be evaluated.

The adaptation method has been a success in that the company has come up with a simple process for their most common project type, which has been made available for all employees. Whether this process becomes a success will be determined through further studies of the actual use patterns.

Further Research

Adoption of RUP: Figure 1 shows some possible ways of tailoring RUP at different levels in a software developing organization. In this case study we have been following an organization which chose the project type adoption.

It is of interest to also follow more closely organizations selecting an organizational adoption, or a project adoption. The success and failure criteria in each case should be compared and analyzed.

Experiences from use of tailored RUP: In this case we did follow the process of tailoring and partly, documenting, a project type tailored RUP. We cannot say for sure if the tailoring has been successful until we have empirical results from the use of the tailored RUP. The next step in the research together with this company will be to collect experiences from the use of this instance of RUP.

Metrics: What kind of metrics should be applied when we are interested to measure the process of tailoring RUP in different organizations, and done in different ways? What kind of metrics should be applied when we try to evaluate the success of the use of the tailored RUP in different types of projects in different organizations? How to apply metrics when it comes to measure a software process is still an uncovered aspect of software process improvement, and we think that an association to a single process framework, like RUP, may ease the process of defining and validating metrics for software processes.

References

1. Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, ed. A.W. Longman. 1999, Reading: Addison Wesley Longman. 463.
2. Krutchen, P., *The Rational Unified Process: An Introduction*. 2nd ed. 2000: Addison-Wesley. 298.
3. Bergström, S., Råberg, L., *Adopting the Rational Unified Process*. 2004, Addison-Wesley. p. 165-182.
4. Karlsson, F., P.J. Ågerfalk, and A. Hjalmarsson. *Method Configuration with Development Tracks and Generic Project Types*. in *CAiSE/IFIP8.1 International Workshop in Evaluation of Modeling Methods in Systems Analysis and Design*. 2001. Interlaken, Switzerland.
5. http://www-1.ibm.com/support/docview.wss?uid=swg21158199
6. http://www.m-w.com/dictionary.htm
7. Nonaka, I., Takeuchi, H., *The Knowledge-Creating Company*. 1995: Oxford University Press.
8. Avison, D., *Action Research.* Communications of the ACM, 1999. **42**(1): p. 94-97.
9. Susman, G., Evered, R., *An assessment of the scientific merits of action research.* Administrative Science, 1978. **23**(4): p. 582-603.
10. http://www.atlassian.com/

P3: A study of a Mentoring Program for Knowledge Transfer in a Small Software Company

Presented at the 6th International Conference on Product Focused Software Process Improvement, Oulo, Finland, 2005
Available in F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2005, LNCS 3547, pp. 245-256, 2005. © Springer-Verlag Berlin Heidelberg 2005. Reprinted with kind permission of Springer Science and Business Media.

Finn Olav Bjørnson[1], Torgeir Dingsøyr[2]
[1]Department of Computer and Information Science,
Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
bjornson@idi.ntnu.no
[2]SINTEF Information and Communication Technology
NO-7465 Trondheim, Norway
Torgeir.Dingsoyr@sintef.no

Abstract. Mentor programs are important mechanisms that serve functions such as career development as well as knowledge transfer. Many see mentor programs as an efficient, inexpensive, flexible and tailored way of transferring technical knowledge from experts to less experienced employees. We have investigated how a mentor program works in a small software consultancy company, and propose that the learning effect of the program could be improved by introducing methods to increase the employees level of reflection.

1 Introduction

Small software consultancy companies have to leverage their position in the market to stay ahead of their competitors. In order to survive, the solutions provided by their consultants have to be of such quality that makes their customers return to the company when they need assistance with a new project, and the solutions should ensure a good reputation for the company that attracts new customers.

To ensure high quality in the systems developed, companies are dependent on a good software development process. The main parts of this process can be planned out in advance and used collectively in a firm in order to ensure quality, but in every project you will probably run into situations where it is important to be able to improvise in order to keep the project on tracks. This is especially true for small software intensive companies in turbulent environments [1]. In these situations experience play a major role in coping with the different challenges.

Experienced developers recognize many different problems and often know the appropriate solutions straight away. For new developers however, this is often not the case. Also if the company is dependent on remaining agile and changing their process in accordance with the demands of their customers, the experienced developers may loose their ability to see the best solution. In these circumstances a company has to have a good strategy to manage their collective knowledge.

Wickert and Herschel [2] examine various challenges that small businesses face when implementing knowledge management [3] efforts. Small businesses often do not have the time and resources that larger companies have to implement large knowledge management efforts, yet they are more vulnerable to knowledge erosion through leaving of key employees. In such an environment it becomes vital to share knowledge to prevent knowledge erosion and staying up-to-date. One suggested solution is mentoring programs which can have an effect in leveraging personal knowledge and sharing knowledge between projects. Such programs can often be more effective than training and written documentation [4].

In this paper we describe an ongoing research project to improve the mentor program in a small consultancy company. The main purpose of the mentor program in this company is knowledge transfer, particularly concerning the software development process and project management. We have focused on how the mentor program supports learning, and changes that could increase the learning effect of the program.

The organization of the paper is as follows: First, we present theory on mentor programs and learning as a part of mentoring. Then, we present the research approach used in this work. We present a small software company where we have conducted a study on a mentoring program, present findings and results from initial interviews, and our work with improving the program. Finally, we conclude and present future work.

2 Mentoring programs and learning

In this section, we present work from management theory on what a mentor program is, how mentor programs can be designed, and how learning can take place in mentor programs.

Kram [5] suggests that existing theory predicts that effective mentoring should be associated with positive career and job attitudes. In a literature review, Ragins et.al [6] show that empirical studies supports this proposition. They also present results from a survey that indicate that persons in dissatisfying or marginally satisfying mentor relationship express the same or worse attitudes than people not involved in a mentor relationship at all. One of their conclusions is that it is clear that good mentoring may lead to positive outcomes, but bad mentoring may be destructive and in some cases worse than no mentoring at all.

2.1 What is a mentor and protégé?

According to Kram [5], mentors are generally defined as “individuals with advanced experience and knowledge who are committed to providing upwards mobility and career support to their protégé”. A protégé literally means “a person under the patronage, protection, or care of someone interested in his career or welfare” [7]. This is usually a younger employee who lacks experience in one or more fields.

2.2 Formal and informal mentoring programs

According to a literature review of mentoring by Ragins et.al. [6], comparisons of non-mentored and mentored individuals yield the consistent result that individuals with informal mentors report greater career satisfaction, career commitment and career mobility than individuals without mentors. Many organizations have attempted to replicate the benefits of informal mentoring by developing formal mentor programs. Yet formal and informal mentoring relationships vary on a number of dimensions:

Informal mentor relationships often arise through a mutual developmental need, and often spring from mutual identification. The mentor may view the protégé as a younger version of themself and the protégé may view the mentor as a role model. This mutual identification contributes to a closeness and intimacy of the mentor program which is often cited in mentoring literature [5]. An informal mentor program is often unstructured and the participants meet as often and as long as is desired. Such an informal mentor relationship usually lasts between three and six years. The purpose of informal mentoring relationships is often the achievement of long term career goals for the protégé.

In contrast, formal mentoring relationships usually springs from a third party assigning the mentor and protégé to the relationship. This may lead to people entering into these relationships not because of mutual need but to meet organizational standards. Meetings in a formal mentoring relationship is often sporadic or specified in a contract at the start of the program, and their duration is often from six months to one year, much shorter than informal relationships. Because of this short time span, the purpose of formal mentoring is often the achievement of short term career goals.

2.3 Mentoring as a mechanism for learning

We adopt the definition of learning from [7] “to gain knowledge or understanding of or skill in by study, instruction, or experience”

Kram and Hall claim that mentor activities are “prime and untapped resources in creating the learning organization” [8]. Allen and Eby [9] claim that mentors as well as protégés should benefit from a mentoring program including learning about “new technologies” and receiving updates on issues at other levels of the organization. But they also report that there is still a need to empirically examine these issues.

If we look into the literature on work-based learning, we find much work on the use of public reflection for learning [10]. Reflective practice can briefly be described as thinking about thinking, which is something that should happen in a mentor relationship during discussions.

In theory on learning, Argyris and Schön distinguish between what they call single and double-loop learning in organizations [11]. Single-loop learning implies a better understanding of how to change (or “tune), say a process, to remove an error from a product. It is a (single) feedback-loop from observed effects to making some changes (refinements) that influence the effects, see figure 1.

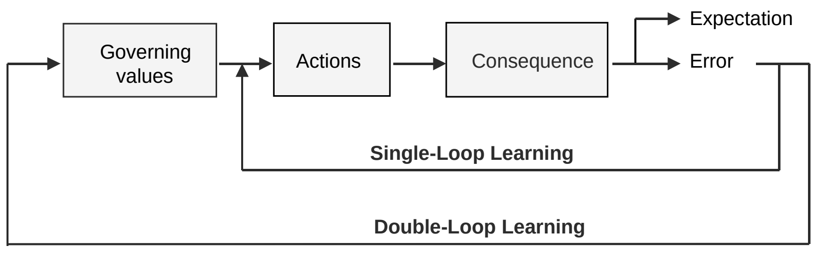

Fig 1. Single and double loop learning.

Double loop learning, on the other hand, is when you understand the factors that influence the effects, and the nature of this influence, which is called the “governing values”. This could be to understand why a process is usable, that is: Which premises must be satisfied for it to be worthwhile. To make changes based on this type of understanding will be more thorough.

In work-based learning, a mentor program is called a “developmental relationship” [12] where participants typically create learning agendas and action plans. The protégé receives feedback from the mentor, and it is likely easier for the protégé to be confident with the mentor than people representing formal line authority. Raelin [12] report that monthly or twice-monthly meetings between mentors and protégés are common. It is typical to start with an assessment of current practice for example through a 360 degree assessment. During the mentor program, good mentors “emphasizes the need for ongoing reflection and inquiry”. When the protégé uses new knowledge in practice they will reflect on the application introspectively and with their mentor. An advice in mentor meetings is that the mentor asks open-ended questions, which might begin with “tell me a little more about your thinking behind that” [12]. This type of discussion can lead to discussion about governing values that lead to decisions, and thus move the learning from single-loop to double-loop.

3 Research Approach: Studying A Mentor Program in a Small Software Company

This research was carried out in a small software consultancy company, which currently employs 50 people, 30 at their main office and 20 at a branch office, located in a different city. Their main source of income comes from three different activities: hiring out developers for pure software development, developing complete solutions for customers and renting out senior personnel as strategic advisors in project management. They have concentrated their customer profile to the domains of healthcare, energy, trade and industry.

One of the main internal goals for this company is to "improve internal knowledge management through revised work processes and *internal training of employees in new processes*". Through our common involvement in a software process improvement research project, we agreed to take a closer look at their mentor program.

We used action research as our research approach because the company was interested in improving practice. Avison et. Al [13] describe action research as "unique in the way it associates research and practice. Research informs practice and practice informs research synergistically. Action research combines theory and practice (and researchers and practitioners) through change and reflection in an immediate problematic situation within a mutually acceptable ethical framework."

We have used an approach in five phases, which are iterated [14]: diagnosing, action planning, action taking, evaluating and specifying learning. This report sums up our work from the initial diagnosing-phase to the action-taking-phase, and details the findings and experiences we have made so far.

For the initial diagnosing phase, we used semi structured interviews. We interviewed six employees, two had acted as mentors, two had been protégés and two had never been involved in the mentor program. The interviews were carried out using an interview guide. All of the interviews were taped using a dictaphone and were subsequently transcribed and sent back to the interviewees for approval and clarification. The material was coded and analyzed using the constant comparison method [15] and the NVivo tool[1].

For the action-planning phase we started with a literature survey of research and management literature concerning mentoring. This was summarized in an internal note to the company. We then held a meeting to discuss the findings from the literature and how they compared to the findings in our interviews.

We are currently in the action-taking phase. We have conducted a workshop with several employees where the goal was to arrive at a new and improved mentoring approach based on the interviews and the research literature. The document detailing the official mentor program has been rewritten to reflect the findings in our research and the outcome of the workshop.

We are currently awaiting projects where the new mentor program can be tried out in practice. When new projects are launched, the researchers will have regular contact with the mentor and protégé and based on interviews with the participants we will evaluate the new approach, and discuss common learning points.

4 Mentoring in a small software company

When we started our research on the mentor program, we got access to documentation that described the existing program. Two 1,5 page internal company memos described the mentoring program, one for the competency area of Rational Unified Process and the Unified Modeling Language, and one for Project management.

When interviewing employees about the mentor program, we discovered several adopted mentor schemes, we were able to gauge the employees’ attitude towards the program, and got several suggestions for improvement.

[1] A tool from QSR International: http://www.qsrinternational.com.

4.1 The existing official mentor program

The purpose of the mentor program was to “spread knowledge and experience to everybody in the company”, by “providing knowledge to projects and persons”, “offer resources and champions [to projects]”, and “offer practical experience in addition to theoretical knowledge”. The mentor program should:

- Make RUP/UML and project management knowledge available for both projects and individuals
- Offer resource persons and initiators
- Offer practical experience in addition to theoretical knowledge
- Offer “controllers” who ensures correct use of RUP/UML/project frameworks in projects
- Offer the consultants “expert support”
- Increase the motivation of employees to use RUP/UML/project frameworks

The mentors were supported by project funds, and it was the project manager’s responsibility to decide on the type and degree of effort of mentoring. The line management then assigned a mentor based on the requirements from the project manager. For large projects, it was written that “a mentor typically should use 1-2 days a week” to solve problems in the design phase. In smaller projects, the effort could typically be two days in the start-up phase, and then 2-4 hours a week thereafter.

4.2 Different mentor schemes

Even though we had been sent to investigate how the current official mentor program worked, we quickly discovered that the program was not that well known: "*I know very little about the formal mentoring program*", "*I do not know of it and do not know what it entails. So if we have this program we have not gotten any information about it*".

In addition to the official mentor program, we discovered several unofficial mentoring schemes that had been adopted. The one that most people mentioned was that the entire company functioned as a large network where there was no problem dropping by your colleagues for help: "*We have this kind of informal [mentoring] - the company functions as a large network. If you are working with a project and run into problems, there are always people who have worked with this problem before, and you can use them for support!*" This unofficial mentor scheme seemed to be mostly related to technical problems, but there was some degree of design and analysis problems being passed around too. In contrast, the official mentor program was mostly related to the software development process and project management.

The most important factor to keeping the informal mentoring scheme alive seemed to be various social initiatives in the company: *“I think the most important thing, what works best, is gatherings and such. Where you get away from the office. You talk, and get to know people and what they are doing. A lot of our colleagues are out [at customer sites] and you do not have much contact with them. So it’s a good place to catch up on what they are doing. So after these gatherings its more easy to know where to go to get information that is important to you.”*

Another scheme adopted, was that when they staffed new projects, they always tried to put at least one experienced employee on the project who could act as a kind of mentor to the others. "*Whenever we get a new project and have to staff it - Then it is important that we put someone with experience there, one who has done similar projects before – that way it becomes a kind of mentoring.*"

In addition we discovered a program designed for new employees where they got a “sponsor” the first month after they were employed. The sponsor was responsible for showing them around and introducing them to the company. In that way it was kind of an introduction to the unofficial mentor scheme. *“It is mostly routines. How things are done here. Practical info to get you started – but we do have a greeting round, where you meet everyone and they tell you about what they are doing. In that regards it could be seen as an introduction to it [informal mentoring]”*

We also discovered that they already had a formal approach to mentoring, which they used when they hired out consultants as mentors, but this degree of formalism was seldom used in-house. "*When we are in the market and hire out consultants. Then it is clear that, ok this is mentoring, and that makes it a lot more formal. We try to do it in-house too, but it is much more formalized when we offer it in the market!*"

4.3 Attitudes to the mentor program

Even though the official mentoring scheme was little known in the company everyone we interviewed was invariably positive to having such a program. However the comments varied with the degree of involvement in the program.

The people who had not used the mentoring program commented that it would be nice to have access to such a program: “*I see it as a great advantage if we could do it that way*”, “*concerning process, we have a lot of knowledge in the company about that, it should be easy to create programs where the experts can help out in different situations*”

The people who had used the mentor program also commented on the importance of having the mentor program and on the positive effect of having a mentor to talk to about different solutions to a problem: “*It was quite nice to have someone you could turn to and consult about different approaches to a problem.*”, “*mentoring is absolutely positive. It is important that we have this program.*“

The people who had functioned as mentors were also positive to the program but their comments were more concerned with the benefits of the program: “*It improves our level of competence … we get to discuss our profession, because it is a lonely role [project manager], especially when we are hired out and are at the site of a customer. … The internal communication improves*”, “*It acts as a kind of quality control … it helps us deliver a better product to our customers”, “I think it makes people feel safe, safe in that they are not alone in their jobs. You create a transfer of competence and you create a relation between the two that can be used later on*”

4.4 Possible improvements to the mentor program

During the interviews, the employees were also asked for suggestions on what could be done to improve the mentor program. Again the response varied according to level of involvement.

Those who had not been involved in the mentor program so far saw the need for more formalization on the routines of getting a mentor. "*We should have a checkpoint in the start-up routines of a project. You do not necessarily have to use a mentor, but you should at least make a conscious choice!*" That being said they were also concerned that it should not be too formalized. Another concern was how protégés were viewed in the organization, that it should not be considered a sign of weakness to ask for a mentor. They were also concerned for the people acting as mentors. They felt that a mentor should be prepared to accept the job voluntarily.

The employees who had used the mentor program were also concerned with the degree of formalization surrounding the program. "*I do not think the program is formalized enough. It is up to the individual to ask for it. And then – it becomes a limitation on who asks for it and who does not.*" Among the other things they mentioned was that the program was not marketed enough and they felt the need for a more concrete framework and guidelines concerning the program.

One employee who had acted as mentor saw the need for more formalization in that a lot of potential interesting information and experience was lost in the current program "*It has potential for improvement in that we could try to make it more formalized. Then it would be easier to collect the experience resulting from the different instances … So it can benefit more than just the two…*"

Another mentor mentioned that the interest in a mentor was greatest at the start-up of a project, and then the contact gradually dwindled as the project progressed. He felt that this could be explained by the fact that those who received the mentors help felt more and more confident as time went by, but on the other hand it could also be a bad thing since he felt that it was usually once the projects were well under way that the real problems emerged that experience could help a lot to relieve.

During the interviews we also got the impression that the learning in the mentor program consisted mostly of practical help, or as we saw it single looped learning. There was not a lot of discussion and reflection taking place in the program. "*It was mostly assistance with practical things, to get us started. To get started with the right procedures. Get the accounting going, how to keep track of income and so on*"

4.5 Main conclusion from the interviews

After analyzing the interviews, we presented the results for the company to get feedback and to see if they had any comments. Our main conclusion was that most of the learning in the mentoring program that took place seemed to be single looped, and that the company could benefit from trying a double looped approach. There seemed to be confusion about what the mentoring role should contain. What was the difference between mentoring, sponsoring and quality assurance work? There was also disagreement on how formalized the mentoring program should be and what areas it should cover.

5 Improving the mentor program

To improve the mentor program, we held a workshop with the people responsible for the program in which we revised the program based on input from the interviews and research literature. The workshop had the following agenda: short presentation of the results from the interviews, a brainstorm on what the main elements of the mentoring program should be, discussion concerning what separated the mentoring program from quality assurance and the sponsor program, and finally how the mentoring program should be facilitated in order to maximize learning. Before the workshop all participants got a copy of the main findings from the interviews and a short memo on mentoring based on findings in the research literature.

5.1 Important elements of the new mentor program

The first brainstorm session consisted of the company's representatives writing down what they thought important about the mentoring program on yellow stickers and then grouping them together on a whiteboard. This resulted in eight groups of elements that should be considered important in the new mentor program:

- Mutual trust and confidence was stressed as important in order for the program to work, no one should feel threatened by the new program.
- The communication between the participants should be discussion-based in order to better facilitate learning. The mentor should act as a discussion partner and ease learning, not provide direct answers.
- There should be a certain amount of time set aside for mentoring and regular meetings should be scheduled in order to keep regular contact
- Mentoring should be available both on project management and on more technical subjects.
- There should be mutual feedback between mentor and protégé and the mentor should be proactive and not just wait for questions from the protégé.
- The mentor should be funded by project budgets if the projects were large, otherwise he could be supported by the company directly. Not all projects should necessarily have a mentor, but all projects should have the option of using one.
- Mentoring should be initiated both from project management and by whoever felt the need, it should not necessarily be narrowed to one on one consultations.
- Finally the need for all participants to be constructive was stressed.

5.2 A clear separation of roles

The second part of the workshop was to determine where the separation between mentoring, quality assurance and the sponsor program was. They decided on the following separation of concepts:

A quality assurance employee makes sure that all the formal bits are done right. He usually appears later in the project. In small projects the project manager is responsible for this role, in larger projects they can create their own quality assurance role. The quality assurance is a check that the individual has done a good job. This was consistent with their old view on the quality assurance role, the news here was that a mentor should not have the responsibility of this role even though a lot of mentors had done so in the past.

A sponsor is responsible to introducing new employees to the company's routines and provides a social contact point. They do not have regular meetings and are not responsible for questions concerning the profession. A sponsor is only provided for new employees for the first month in the company. This was also consistent with their old definition. From our findings in the interviews we also suggested that this role should be conscious on helping the new employee get familiar with the unofficial mentoring scheme.

A mentor makes sure that the project is carried out professionally by providing expertise and advice. The need for mentors was usually greater in the start of a project. The mentor provides the individual with the help to perform a good job. Furthermore it was decided that the official mentor program should be split into three parts:

- Non-formal mentoring: As we discovered in the interviews, this was already taking place and the environment was supportive of such a scheme. It functioned in an ad-hoc manner and it was decided that this should continue to function since it was obvious that it was working well. To further support this kind of mentoring it was suggested to invest in social initiatives to keep and improve this environment.
- Formal mentoring should continue more or less as it had functioned, but with few modifications. Basically the important elements defined above were taken into consideration. It should be project based, could potentially be one mentor who would work with several protégés, and be more discussion based.
- They also introduced a new type of mentoring, the trainee program. This was a new role in the company, and much more practical oriented than the previous mentor program. The idea was to introduce employees to new domains (business, technical or management) by allowing employee to follow seniors out to customers and letting them participating in customer meetings and project activities. In practice everyone could go into a trainee role to learn a new domain or a new role like project manager or learning how to handle customer relationships.

5.3 How to improve learning in the mentor program

The final discussion in the workshop was around the problem: How can we improve learning in the mentoring program. This resulted in seven main elements that could be considered by the mentors in the company:

- The mentors should to a large degree post open questions in order to make the protégés think for themselves.
- Confidence and trust between the participants was considered important in order to facilitate learning, this would to some degree be dependent on personal chemistry, but could also be facilitated by patience on both accounts, the ability to pick up signals, the ability of protégés to dare to ask ”stupid questions”.

- A mentor leading a group of protégés could also be considered; the more people the more discussions.
- The mentor should mainly explain and advise by giving examples of how thing had previously been done.
- A good mentor should allocate time, discuss their expectations and provide good feedback.
- A good protégé should realize their needs; this could be facilitated by better information from the company.
- It was also considered important to learning to have a clear definition of roles.

The main discussion points of the workshop was written into the memo describing mentoring in the company, which was extended from one and a half pages to three pages.

6 Conclusion and Future Work

We have investigated a mentor program in a small software consulting company in order to identify issues that could be improved. We found many different mentor schemes to be in place in the company, found arguments in favor and against a more formal approach to mentoring in the company. We found most of the learning that took place to be single looped. In order to increase the learning effect, we discussed how we could introduce more reflective practice into the mentoring program, and identified some efforts that were taken into a revised mentoring program. We also made a clearer separation of roles, and suggested that mentoring should have a greater availability in the company.

We believe that the new mentoring program will provide better support for double loop learning through increased reflection. The amount of reflection should increase when the mentors pose more open questions during meetings. Also, organizing mentoring in a group of protégés should lead to more discussion, which should also lead to more reflection on current work practices.

The new mentoring program has been introduced through a meeting with all employees, and now that the work of restructuring the mentor program is done, we switch to an observer role. We will follow mentor and protégé pairs in new projects and evaluate the changes brought on by redefining the mentor program. By performing the same interviews again on people using the new mentor program, and by observing how the new program runs over time, we hope to be able to ascertain how successful the knowledge initiative have been for the company, and how it influences their software development process.

Acknowledgement

This work was conducted as a part of the Software Process Improvement through Knowledge and Experience (SPIKE) research project, supported by the Research Council of Norway through grant 156701/220. We are very grateful to our contact persons in the software consulting company for providing stimulating discussions and help with organizing the work.

References

1 T. Dybå, "Improvisation in Small Software Organizations", IEEE Software, no. 5, vol. 17, pp.82-87, 2000.

2 A. Wickert and R. Herschel, “Knowledge management issues for smaller businesses", Journal of Knowledge Management, no. 4, vol. 5, pp. 329-337, 2001.

3 M. Lindvall and I. Rus, “Knowledge Management in Software Engineering”, IEEE Software, no. 3, vol. 19, pp. 26-38, 2002.

4 F. J. Armour and M. Gupta, “Mentoring for Success”, IEEE IT Pro, no. May - June, pp. 64-66, 1999.

5 K. E. Kram, Mentoring at work: Developmental relationships in organizational life. Glenview, IL: Scott Foresman, 1985, ISBN: 081916755X.

6 B. R. Ragins, J. L. Cotton, and J. S. Miller, “Marginal Mentoring: The Effects of Type of Mentor, Quality of Relationship, and Program Design on Work and Career Attitudes”, Academy of Management Journal, no. 6, vol. 43, pp. 1177-1194, 2000.

7 Webster's, Encyclopedic Unabridged Dictionary of the English Language. New York: Gramercy Books, 1989.

8 K. E. Kram and D. T. Hall, “Mentoring as an antidote to stress during corporate trauma”, Human Resource Management, vol. 28, pp. 493-510, 1989.

9 T. D. Allen and L. T. Eby, “Relationship Effectiveness for Mentors: Factors Associated with Learning and Quality”, Journal of Management, no. 4, vol. 29, pp. 469-486, 2003.

10 J. A. Raelin, “Public Reflection as the Basis of Learning”, Management Learning, no. 1, vol. 32, pp. 11-30, 2001.

11 C. Argyris and D. A. Schön, Organizational Learning II: Theory, Method and Practise: Addison Wesley, 1996.

12 J. A. Raelin, Work-based learning. Upper Saddle River, NJ: Prentice Hall, 2000.

13 D. Avison, F. Lau, M. Myers, and P. A. Nielsen, “Action Research”, Communications of the ACM, no. 1, vol. 42, pp. 94-97, 1999.

14 G. Susman and R. Evered, “An assessment of the scientific merits of action research”, Administrative Science Quarterly, no. 4, vol. 23, pp. 582-603, 1978.

15 M.B. Miles and A.M. Huberman, Qualitative Data Analysis: An expanded sourcebook, second ed. SAGE publications, 1994.

P4: Defining Software Processes Through Process Workshops: A Multicase Study

Presented at the 8th International Conference on Product Focused Software Development and Process Improvement (PROFES'2007)
Available in J. Münch and P. Abrahamsson (Eds.): PROFES 2007, LNCS 4589, pp. 132-146, 2007. © Springer-Verlag Berlin Heidelberg 2007. Reprinted with kind permission of Springer Science and Business Media.

Finn Olav Bjørnson[1], Tor Stålhane[1], Nils Brede Moe[2], and Torgeir Dingsøyr[2]

[1]Department of Computer and Information Science,
Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
{bjornson, stalhane}@idi.ntnu.no
[2]SINTEF Information and Communication Technology
NO-7465 Trondheim, Norway
{Nils.B.Moe, Torgeir.Dingsoyr}@sintef.no

Abstract. We present the application of the process workshop method to define revised work processes in software development companies. Through two empirical action research studies, we study the impact of company premises and goals on the execution and subsequently on the results of the method. We conclude that both premises and goals will influence the workshops, and suggest how the focus of the workshops should be altered to achieve better results depending on the context. We also strengthen previous claims that the process workshops are a good arena that fosters discussion and organizational learning, and that involvement in the workshops leads to higher acceptance and usage of the resulting process.

Keywords: Software Process Improvement, Project Workshop, Empirical Study, Action Research

1 Introduction

The way we develop and maintain software, or the software process, has long been regarded as crucial for software quality and productivity [16]. In many companies, software development is performed in a rather informal fashion, and problems of late and unsatisfactory deliveries are not uncommon.

Problems related to the use of informal development include problems with transferring competence from one project to another, difficulties in establishing best practices, and the widely varying nature of problems to be solved. In order to address these challenges and to improve the quality of the software development process, a lot of companies develop process guides to structure their work.

The process workshop (PWS) method was designed as a lightweight method to help facilitate the development of such process guides. Apart from the original introduction of the process workshop [11] and a Finnish application of the same method [19], there is little empirical evidence on the practical application of this method. This paper aims to add to the body of knowledge on process workshops as a tool for software process improvement, and describes how company context and goals affects the execution of the method and its results.

In the following we describe our work in two companies, hereafter referred to as Alpha and Beta Company. One is a small and one is a medium sized software company, and they both used process workshops to define their software process. Our focus is on the process workshop itself and how processes were constructed. The description of this process, i.e., how it will later appear in an electronic process guide, and the cost-benfit of the process workshop method is as such outside the scope of this paper. Our research goal which we want to answer in this paper is:

How do available information, company context and goals affect the execution and results of process workshops?

The paper is structured as follows: In chapter 2 we take a closer look at related work, and the method we adapted for our cases. Chapter 3 describes the research method employed in each case. Chapter 4 gives a deeper introduction to each case. Chapter 5 discusses the differences between the cases and our findings. Chapter 6 concludes our findings and describes possible routes for further research.

2 Related Work

When companies choose to design their own development processes, one option is to assign the task to a group of expert “process engineers” as described by Becker-Kornstaedt [7, 8]. One or more process engineers elicit process data from interviews, documents, surveys, e-mails and observation, and then interpret this data to produce a process model. This approach relies heavily on the experience and skill of the process engineer. Therefore, without any structured method, quality and repeatability cannot be ensured. It is, however, unlikely that the use of qualitative methods alone can compensate for experience in process modeling and software engineering [8]. When using a process engineer to formulate a process model, it is common to create a descriptive model. A descriptive model is a model, which expresses processes currently in use. Descriptive software process modeling is an important part of any software process improvement (SPI) program, because descriptive modeling allows process engineers to understand existing processes, communicate process and analyze existing practices for improvement [8]. For this reason, much work has been done on proposing languages, techniques and tools for descriptive process modeling.

An alternative to using process engineers is to involve the employees more in designing the process models, for example through workshops [1, 17]. This type of work takes up the heritage from employee participation in organizational development, a part of Scandinavian work tradition as well as in most work on improvement, from the Total Quality Management principles [10] to the knowledge management tradition in

Communities of Practice [25]. Participation is also one of the most important foundations of organization development and change [17], and one of the critical factors for success in software process improvement [13].

Some studies have found that employee involvement lead to organizational effectiveness, measured through financial performance, turnover rate and workforce morale [21, 24]. Another potential effect of participation is increased emotional attachment to the organization, resulting in greater commitment, motivation to perform and desire for responsibility. Riordan et al. [21] use a framework with four attributes to define employee involvement:

- Participative decision
- Information sharing
- Training
- Performance-based rewards

There are several techniques available for achieving participation. Examples are search conferences [20], survey feedback [6], autononomous work groups [14], quality circles [14, 15]. All of which are predicated on the belief that increased participation will lead to better solutions and enhanced organizational problem-solving capability.

In software development, the software developers and the first-line managers are the ones who are into the realities of the day-to-day details of particular technologies, products, and markets. Hence, it is important to involve all who are part of the software process, and have decisions regarding the development of process guides made by those who are closest to the problem.

Consequently, and in order to get realistic descriptions with accurate detail as well as company commitment in an efficient manner, all relevant employee groups should be involved in defining the processes. This can be done by arranging several process workshops [17] in the form of quality circles [15] as a tool to reach a consensus on work practice. A quality circle is composed of volunteers who arrange regular meetings to look at productivity and quality problems, and discuss work procedures [15]. The strength of the circle is that they allow employees to deal with improvement issues that are not dealt with in the regular organization. The quality circles used in the process workshop have all been temporary, and created with a relative well-bounded mandate to be fulfilled. Once a sub-process has been accomplished, the circle is disbanded. This kind of quality circles is also known as “Task forces” [14].

2.1 The Process Workshop Method

In the studies reported in this paper, we used a method called process workshop [11]. The method is designed to involve the users of the future process in discussing and defining the processes. It ensures that people discuss how they work – which fosters learning even before the process guide is available in the company. It also assures quality – the process guide is developed by people who know how to do the work; it does not describe how external consultants or senior staff imagine what “ideal” development processes should look like.

The process workshop approach to defining process(es) consists of six main steps and five sub-steps as shown in Figure 1 below. Since the focus of our work is on the process workshop itself, we only provide details of the relevant substeps here. More details on the process workshops method can be found in [11].

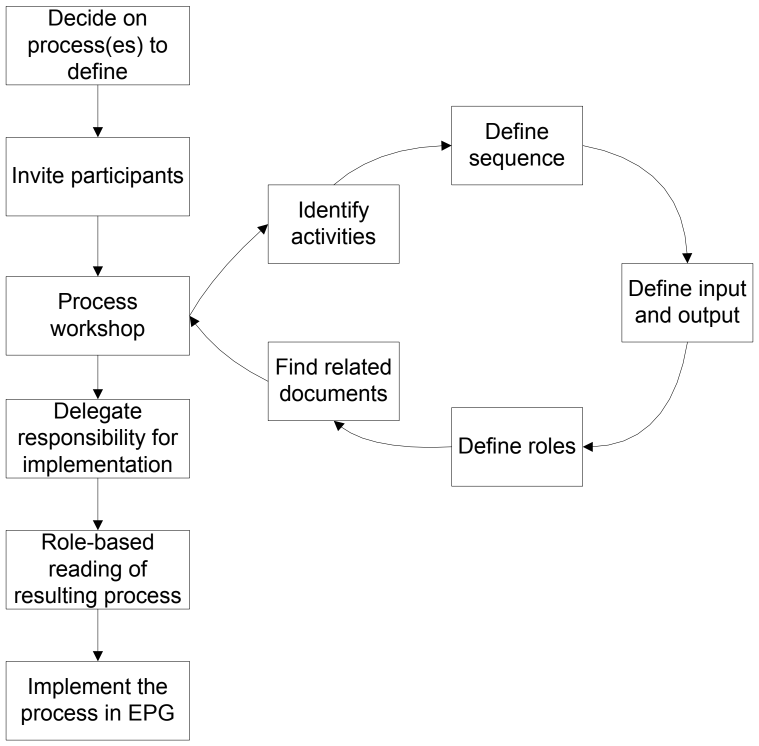
Decide on process(es) to define
Invite participants
Process workshop
Delegate responsibility for implementation
Role-based reading of resulting process
Implement the process in EPG
Identify activities
Define sequence
Define input and output
Define roles
Find related documents

Fig. 1. Steps to define process in a workshop

The theoretical approach of the five sub steps are:

- *Identify activities.* Brainstorming on the main activities of the process by using the KJ process [22] and documenting the result. The KJ is a creative group technique to organize and find relations between seemingly unrelated ideas.
- *Define the sequence of the activities.* A suitable workflow between the activities from the previous phase is found.
- *Define inputs and outputs.* Identify documents or artifacts that must be available to start a given sub-process, and which documents that mark the end of such sub-processes. Conditions that must be satisfied to begin or exit the sub-process can be described in checklists.
- *Define roles.* Defining which roles should contribute in each activity.
- *Related documents.* Identify documents that either already exist in the company, or new documents that would be helpful in carrying out the activities. Such documents can be templates, checklists and good examples of input or output documents.

A process workshop can be used both to make a descriptive process model and to directly formulate a new and improved process. In the latter case process models are improved directly in the workshops through the discussions, without an analysis of the present situation.

3 Research Method

This study reports on two separate empirical studies. Each study investigated the application of process workshops to define software processes for software development companies. However, the research method differed slightly between the two cases, and the companies are also at different stages in their improvement efforts. The research method and the difference in application to the two companies are described in this chapter. Two of the authors of this paper were responsible for the research at the Alpha Company, while the two others handled the research at the Beta Company.

Both Alpha and Beta were involved in the same national research project, aimed at investigating software process improvement in software engineering. Due to the cooperative nature of this research project, the research method adopted for both companies was the participative research method, action research [4]. In order to properly describe and differentiate the research methods used, we describe them according to the five principles suggested by Davison et al. [9] (table 1) and the three aspects of control structures suggested by Avison et al. [3] (table 2).

Table 1. The five principles of canonical action research, by Davison et al.

	Principles of canonical action research
1.	The principle of the researcher-client agreement.
2.	The principle of cyclical process model.
3.	The principle of theory.
4.	The principle of change through action.
5.	The principle of learning through reflection.

Table 2. Forms and Characteristics of the major AR control structures, by Avison et al.

Control aspect	Forms	Characteristics
Initiation	Researcher	Field experiment
	Practitioner	Classic action research genesis
	Collaborative	Evolves from existing interaction
Authority	Practitioner	Consultative action warrant
	Staged	Migration of power
	Identity	Practitioner and researcher are the same person
Formalisation	Formal	Specific written contract or letter of agreement
	Informal	Broad, perhaps verbal, agreements

Evolved	Informal og formal projects shift into the opposite form

At the Alpha company, the research on how to use project workshops to define software process was carried out during 2003. The process was later implemented in an electronic process guide, and the use of the guide over time was studied [17]. The research on project workshops to define their software process at Beta Company was carried out during 2005, in other words after the study at Alpha company. Since the goal of the company was close to that of Alpha, we decided to adopt the method of process workshops to define the process. The company wanted to define their process, and the researchers got a chance to empirically evaluate the method previously suggested and used at Alpha.

Concerning the first principle of researcher-client agreement, this research was done in a general project on software process improvement, where both companies wrote an improvement plan and the researchers wrote a research plan for each company.

The research followed the action research model (principle two) proposed by Susman and Evered [23] in discussing the situation at the companies, planning action, taking action, evaluating action, and finally specifying for learning. The research has gone through three “evolutionary” cycles at Alpha, however our focus for this paper is on the first cycle in which the process workshops were held to establish the process. At Beta we have only done one evolutionary cycle at the present time.

The third principle of theory, was satisfied for both companies through the research questions and our focus on developing and testing the method based on the theory of user involvement [14, 15, 21, 24].

The fourth principle of change through action was satisfied through the actions of holding the project workshops. The results form the basis for a new electronic process guide, which includes examples based on the defined process. These results have been used to implement the new defined process at Alpha, whereas Beta has not come this far yet.

The fifth principle of learning through reflection was achieved at Alpha through project meetings in which the researchers and company representatives discussed actions that were taken and analyses made by the researchers. At Beta the results were discussed in a series of meetings, we held a post mortem analysis (PMA) [23] of the project workshops to evaluate it at the end, and conducted an interview with the person responsible for the process improvement initiative at the company.

From the aspect of control structures on action research, we can put the following characteristics on the research projects. The initiation was collaborative for both projects. Both the company and the researchers were in a common research project aimed at improving software processes, and the research plan was developed from the joint wishes of practitioners and researchers.

The authority of the projects is where we observe the main difference. At Alpha it is characterised as staged. In the beginning, the researchers were heavily involved with developing the workshops, while the company assumed more of the responsibility and workload towards the end. At Beta we also characterise the authority as staged, but the oposite effect was seen. In the beginning, the company was very much involved with developing a solution, but as an external project demanded more and more of their resources, power was transferred to the researchers who had to carry much of the workload.

The formalization of both projects can be said to have evolved from formal in the beginning, with a clear structure and plan, to more informal at the end.

4 Empirical results from the two software companies

In this chapter we describe the two companies in which we conducted our research in greater detail. We describe the context, the practicalities surrounding the process workshops, how the companies used the data from the workshops, and finally an evaluation of the workshops themselves.

4.1 Alpha Company

Alpha Company was founded in 1984, and is one of the leading producers of receiving stations for data from meteorological and Earth observation satellites. The company has worked with large development projects, both as a prime contractor and as a subcontractor. The company has approximately 60 employees, many with master’s degrees in computing science, mathematics or physics.

The size of typical product development projects are 1000-4000 work hours. Customers range from universities to companies like Lockheed Martin and Alcatel to governmental institutions like the European Space Agency and the Norwegian Meteorological Institute. Most of the software systems that are developed run on Unix, many on the Linux operating system. Projects are managed in accordance with quality routines from the European Corporation for Space Standardisation and ISO 9001-2000 [5].

The company had an extensive quality system which was cumbersome to use because of the size and existence partly on file and partly on paper. Since it also did not emphasize such aspects as incremental and component development, the QA system came under increasing pressure to change. It became impossible to follow the standards and even more impossible to do effective quality assurance work in the projects. As part of being certified according to ISO 9001-2000, the company decided to develop a process-oriented quality system [18].

Defining new processes

Management of the project for defining the new processes was kept with the Quality Assurance (QA) department. One of the two persons working in the QA department had earlier worked as a developer and was now member of the top management. This way this project was anchored both among the developers and managers.

In an initial workshop with both developers and managers it was defined that the process descriptions had to:

- Reflect the “best practices” currently used within the company (take the best from the earlier system into the new system).
- Comply with modern methodologies like the Unified Process and Component Based Development.
- Integrate the process descriptions with important tools for development (e.g. requirements definitions and use-case description).
- Be easy to tailor when a new project is started.
- Be released when the first processes are defined, so it becomes possible to give instant feedback and then keep up the involvement

From these requirements it was decided that the new processes should be created based on “best practice” in the company, with important input from the existing system and engineering tools. It was never an option to first analyse the existing processes and then improve them. This was because they wanted to get the new processes defined quickly to meet the new ISO standard, and to use as little time as possible to keep up the enthusiasm among the developers. The process workshop also provided the possibility to discuss and improve today’s working processes without a thorough analysis. It was also decided that the process descriptions were going to be developed in “process workshops” to achieve participation.

After the requirements were defined, seven process workshops were arranged. Alpha identified four main project types, and they chose “Product Development” - the most common one - as a starting point for the subsequent process workshops. “Product Development” was divided into four sub processes: “Specification”, “Elaboration”, “Component Construction” and “System Integration”.

More than 20 people (1/3 of the staff) participated in one or more workshops. The people who participated in the process workshops were selected by the quality department to represent a variety of roles, experience and opinions. The workshops usually lasted half a day.

Each workshop started by defining the sub-processes in the main process. Then we defined each sub-process activities and their sequence. We used the KJ process [22] for brainstorming and documenting the result. The KJ is a creative group technique to organize and find relations between often seemingly unrelated ideas. After the activities were identified and organized in workflows, the documents for input and output to the process were defined. These documents could be already existing templates, checklist and good examples. Next we identified related roles to each process. After all the sub-processes were defined, the responsibilities for implementing the processes into the electronic process guide.

Implementing the processes

The implementation was executed by QA personnel in a self-made tool and released on the intranet. The first prototype was ready after only a few weeks, and even though the process guide was incomplete it was possible to start real-life testing with a few projects. The projects were encouraged to respond immediately to the process descriptions if they are unclear, uncompleted or unusable. In this way the users were still involved in developing the process descriptions.

The company used 180 work hours in workshops and 1049 work hours in total for development of the first version of the process guide.

4.2 Beta Company

The Beta Company has 20 employees. Their main activities are hiring out consultants as developers, developing complete solutions for customers, and hiring out consultants and project managers as advisors for selecting technology, strategy or process. Typically, no more than four to five consultants are at any time working for the same customer.

The managers of the company wish to leverage the company in the market by providing solutions to the problems of their customers. The solutions should make them stand out and increase the probability that the customers later return with new projects. In order to do this, they wish to foster an environment were all ideas and knowledge are shared freely among the employees, and where the employees can draw upon the experience of each other to provide good services to their customers. This work is difficult since a lot of the employees at any given time are out at the customers' site where they don't have direct access to their colleagues.

One of the identified stumbling blocks for experience sharing and reuse was the lack of a common process and a common set of document templates. In order to remove, or at least reduce this problem, the company wanted to define, document and implement a framework that could be used for development, consultancy and operation. The framework should be easily accessible for all employees and should help them to do their jobs better than today and to show Beta as a highly competent consultancy company. The company started to drift away from this goal after approximately six months and decided instead to document how they worked now. A shift from prescriptive to descriptive modeling. Although never explicitly stated, the focus was on identifying the documents – artifacts – that were produced, who produced them and how. In addition it was important for the company to create an awareness of and understanding for the use of a process that encompassed all development activities. At present, the developers thought in terms of jobs – things to do – not in terms of processes and artifacts. One of the goals was to make them think and work in terms of processes and process steps.

Defining new processes

When the researchers became involved, we saw it as a good oportunity to further test the process workshop method to document their process. We used a sequence of process workshops – one for each of the identified main processes that the company used. The input to the workshops was mainly the developers' experiences with the way they had worked in previous projects. Since the company had no single, defined process and each project more or less invented its own, this was a quite diverse source of information and experience. Each participant brought with him experiences from several processes.

Since part of the goal of the Beta Company was to see which artifacts were needed, we tried to use the standard process worksheet, which has a separate area for documents. However, the workshop participants ignored this area and preferred to mix activities and documents in the same diagram. One of the reasons for this may be that different workshop participants had different ideas about what was done in a project. It was much easier to agree on the documents that are developed than to agree on how they are produced.

We held a total of six workshops over a period of 12 months. Five of the developers participated in two or more of the workshops while an extra five participated in at least one. The workshops treated the processes: requirements, estimation, analysis, implementation, testing and project control and follow-up activities. In addition, we arranged a Post Mortem Analysis (PMA) [12] workshop to assess the whole process workshop series.

We will not treat the results from each workshop in any detail but will instead focus on the workshop process and its results. In addition, we will discuss some of the results from the workshop PMA.

We used the KJ process to create the diagrams during the workshops. Based on the resulting diagrams it was straight forward to see which documents were generated. It is important to note that while the workshop participants were fairly clear on which documents to produce, they are rather vague on the process steps.

Implementing the processes

Even though documents such as use-case descriptions were generated in this process, all of the documents created in the requirements process will resurface in later processes and will be refined there. In the developers' view it was therefore unreasonable to claim that a certain document "belonged to" a certain process or process step. For this reason, the company decided on the following approach to get a unified process concept:

- Identify all documents and code them with information on the process they are generated in and where they later are refined or used.
- Identify all document dependencies, i.e. which documents use which other documents.
- Store templates and examples for all documents that are used in one or more processes.
- Define a discussion tread for each document. This will enable all developers to give input on their experience, what works, what does not and how can we improve on the templates.

Evaluating the workshop approach

When all the company's processes had been analyzed in a process workshop we arranged a PMA to identify strong and weak points in the workshop process used. Most of the negative points related to the lack of participation from the company's management and does not contribute to our understanding of the use of process workshops. The KJ diagram for the positive points is shown below in figure 3.

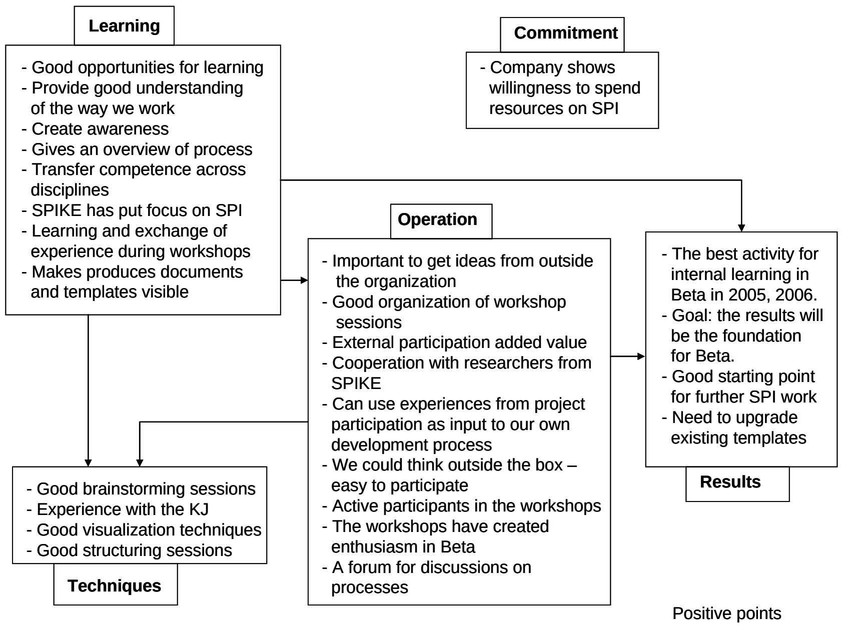
Learning
- Good opportunities for learning
- Provide good understanding of the way we work
- Create awareness
- Gives an overview of process
- Transfer competence across disciplines
- SPIKE has put focus on SPI
- Learning and exchange of experience during workshops
- Makes produces documents and templates visible
Commitment
- Company shows willingness to spend resources on SPI
Operation
- Important to get ideas from outside the organization
- Good organization of workshop sessions
- External participation added value
- Cooperation with researchers from SPIKE
- Can use experiences from project participation as input to our own development process
- We could think outside the box – easy to participate
- Active participants in the workshops
- The workshops have created enthusiasm in Beta
- A forum for discussions on processes
- The best activity for internal learning in Beta in 2005, 2006.
- Goal: the results will be the foundation for Beta.
- Good starting point for further SPI work
- Need to upgrade existing templates
Results
- Good brainstorming sessions
- Experience with the KJ
- Good visualization techniques
- Good structuring sessions
Techniques
Positive points

Fig. 3. Positive KJ diagram from the PMA

Our main experiences can be summed up as follows:

- In a company with many and varied versions of the same process it is easier to focus on documents than on process steps. Dependencies between documents will enforce a sequence of activities but the focus will be on *what*, not on *how*.
- Among the developers, the process workshops are conceived as a positive contribution in several ways, e.g.:
  - Gives an opportunity for active participation - not just asked what you do but be able to use your own experience to contribute to the company's processes.
  - Get a better understanding of the way the company works – an opportunity for learning.
  - External participation – in this case the researchers – added value to the workshops by introducing an outside view on the way the company works

5 Discussion

In this section we discuss our experience with conducting process workshops in different contexts, and elaborate on what we have observed to be the strengths and weaknesses of this approach to software process improvement.

Let us first examine some differences between the two companies and how they chose to employ the process workshop, we have made a comparison in Table 3 below:

Table 3. Comparing Alpha and Beta

Alpha Company	**Beta Company**
Medium sized company 60 employees	Small sized company 20 employees
Mostly in-house projects for external customers	Mostly external projects at customer sites
ISO 9001-2000 certified	No formal certification
Extensive quality system was already in place before the researchers arrived, but it had become outdated and was too cumbersome to use.	No quality system or defined process in place. Each project followed its own process.
Management of the improvement project was handled by a separate Quality Assurance department.	No Quality Assurance department exists, the improvement project was handled by a project manager.
The improvement project had good anchoring with both management and developers through the QA department.	The improvement achieved good anchoring with the developers who participated in the workshops but suffered from poor anchoring with management.
PWS used to define the future process based on best practice. (Prescriptive modeling)	PWS used to understand the current process. (Descriptive modeling)
More than 20 people, 1/3 of the employees, participated in one or more workshops.	5 developers participated in two or more workshops while another 5 participated in one of the six workshops.
Half work-day workshops. ~4 hours	~3 hour workshops after office hours.
Responsibility for documentation of the workshop results was distributed among the participants.	Responsibility for documentation of the workshop results was left to the researchers.
Activity focus in the workshops.	Document focus in the workshop.
Evaluation of the PWS based on researcher observations and observations of the use of the electronic process guide.	Evaluation of the PWS based on researcher observations, post mortem analysis with PWS participants, and interview with the project manager responsible for the SPI effort.

The largest difference between the workshop methods employed in the two companies is the focus of defining future processes based on best practice in Alpha vs. defining the current process in Beta. Originally Beta wanted to define a future process, but given the different processes that emerged through the workshops, it was decided at an early stage to focus on the current processes. In retrospect we can explain the difference in focus with the situation the companies was in at the beginning of the improvement projects.

The employees at Alpha were already used to using a defined process, while Beta had no experience on using a company process. This can also be linked to the project profile in the companies. While Alpha had fairly homogeneous projects, Beta had a heterogeneous profile, with many consultants spread over several external customer sites.

The difference in previous process knowledge also manifested itself in the discussions and subsequently in the results of the workshops. While the employees at Alpha was more comfortable discussing activities, or *how* things should be done, the employees at Beta gravitated towards discussing documents or artifacts, or *what* should be done. That being said, the discussions at both companies kept discussions on the activities of the process to a fairly high level. Neither descended into a detailed description of how an activity should be carried out. The tendency of workshop participants to keep the discussion on a high level is also noted in the study by Pikkarainen [19].

Another result from our two case companies is that management support and involvement is a major success factor. This is nothing new in the literature [13], but we believe it deserves mentioning. At Alpha we had the support of top management through the QA department. At Beta top management was interested, but did not have the time or resources necessary to follow the project. This resulted in other external projects taking precedence over the improvement project. There was also no external drive towards formal certification like there was at Alpha, which could have increased the importance of the improvement project. This can also be explained through Beta's relatively small size, with only 20 employees, putting bread on the table and paying the bills came first. There were not enough resources to dedicate an employee to driving the project. The practical result has been that the researchers have had to provide some of the drive, and the project has taken longer time than anticipated.

Even though there were differences in the premises for the process workshops and slight differences in the execution, both Alpha and Beta employees praised it as a good arena for learning. The project workshops provided an arena where employees from several departments could meet and discuss. This gave the participants a broader view of how work was conducted in the organization. Through this open forum, the employees could discuss and reflect on their own work methods. Not being forced into a new process by external consultants or a distant QA department, creates an arena and opportunity for what Argyris and Schön [2] describes as double looped learning. Pikkarainen et al. [19] also found the workshop approach a good support for organizational learning.

Another effect we observed in both Alpha and Beta was that involvement in the process workshop created ownership of the resulting process. This effect was studied in Alpha, where it was shown that the participants of the workshop used the resulting process guide much more than the employees who did not participate. Although Beta has not implemented the resulting process yet, there have already been indications that there is a difference between those who participated and those who did not.

6 Conclusion and Further Work

We have conducted empirical studies on the application of the process workshop approach in two software companies. Our research question was "*How do available information, company context and goals affect the execution and results of process workshops?*" Based on the results and the previous discussion, we can conclude that:

- The premises of the company will strongly influence the execution of the project workshops. If the employees of a company are used to working according to a process, the workshops can be used to formulate the starting point of a new process based on best practice. If, however, no clear process exists, the focus of the workshops should be on reaching an agreement on the current process before improvement is suggested.
- If the PWS approach is used to reach an agreement on the current process, a good starting point is to focus the discussion on artifacts, or what should be produced, rather than how it should be produced.
- If the PWS approach is used to specify future processes based on best practice, the discussions should be focused towards activities, or how the projects should be run.

In addition to answering our research question, we have made three observations pertaining to organizational learning and some related issues:

- The PWS approach is a good tool for organizational learning. Through the discussions in the workshops, the employees start the learning process, even before the process is available through a process guide.
- Involvement in the workshops fosters ownership of the resulting process, and as such it is a good way to get the developers to actually use the process later.
- The process workshop is a lightweight approach to defining a process for companies. As such it is well suited to small and medium sized companies. It does, however, require some resources to be truly successful and therefore, management support is important.

Further work in this area will be to investigate methods to spread the acceptance and usage of the resulting process. In a previous study [17] we showed that participants had a higher usage level of the resulting process than those who did not participate. In the empirical studies reported in this paper, we had a participant level of about 1/3 of the employees in each company. The challenge now becomes how to get the rest of the employees involved to foster a higher acceptance level of the resulting process.

Acknowledgments. This work was conducted as a part of the Software Process Improvement through Knowledge and Experience research project, supported by the Research Council of Norway through grant 156701/220. We are very grateful to our contact persons in the software consulting company for providing stimulating discussions and help with organizing the work.

References

1. Ahonen, J.J., Forsell, M., and Taskinen, S.-K.: A Modest but Practical Software Process Modeling Technique for Software Process Improvement. Software Process Improvement and Practice. 7(1) (2002) 33-44
2. Argyris, C. and Schön, D.A.: Organizational Learning II: Theory, Method and Practise: Addison Wesley. (1996)
3. Avison, D., Baskerville, R., and Myers, M.: Controlling Action Research Projects. Information Technology & People. 14(1) (2001) 28-45
4. Avison, D., Lau, F., Myers, M., and Nielsen, P.A.: Action Research. Comm. ACM. 42(1) (1999) 94-97
5. Avison, D.E. and Fitzgerald, G.: Information Systems Development: Methodologies, Techniques and Tools. 2nd ed. New York: McGraw-Hill. (1995)
6. Baumgartel, H.: Using employee questionnaire results for improving organizations: The survey "feedback" experiment. Kansas Business Review. 12 (1959) 2-6
7. Becker-Kornstaedt, U.: Towards Systematic Knowledge Elicitation for Descriptive Software Process Modeling. Lecture Notes in Computer Science, 2188. eds. F. Bomarius and S. Komi-Sirviö. Berlin Heidelberg: Springer Verlag. (2001) 312-325
8. Carvalho, L., Scott, L., and Jeffery, R.: An exploratory study into the use of qualitative research methods in descriptive process modelling. Information and Software Technology. 47(2) (2005) 113-127
9. Davison, R., Martinsons, M.G., and Kock, N.: Principles of canonical action research. Information Systems Journal. 14(1) (2004) 65-86
10. Deming, E.W.: Out of the Crisis. Cambridge, Massachusetts: The MIT Press (first published in 1982 by MIT Center for Advanced Educational Services). (2000)
11. Dingsoyr, T., Moe, N.B., Dybå, T., and Conradi, R.: A workshop-oriented approach for defining electronic process guides - A case study. Software Process Modelling, Kluwer International Series on Software Engieering. eds. S.T. Acuña and N. Juristo: Boston: Kluwer Academic Publishers. (2005) 187-205
12. Dingsøyr, T.: Postmortem reviews: purpose and approaches in software engineering. Information and Software Technology. 47(5) (2005) 293-303
13. Dybå, T.: An Empirical Investigation of the Key Factors for Success in Software Process Improvement. IEEE Transactions on Software Engineering. 31(5) (2005) 410-424
14. Guzzo, R.A. and Dickson, M.W.: Teams in organizations: Recent research on performance and effectiveness. Annual Review of Psychology. 47 (1996) 307-338
15. Lawler, E.E. and Mohrman, S.A.: Quality Circles - after the Honeymoon. Organizational Dynamics. 15(4) (1987) 42-54
16. Lehman, M.M. and Belady, L.A.: Program Evolution: Processes of Software Change: Academic Press. (1985)
17. Moe, N.B. and Dingsøyr, T.: The impact of process workshop involvement on the use of an electronic process guide: a case study. 31st EUROMICRO Conference on Software Engineering and Advanced Applications. (2005) 188-195

18. Nilsen, K.R.: Process improvement through development of an extended electronic process guide - from electronic process guide to integrated work tool. EuroSPI 2004. Trondheim (2004)
19. Pikkarainen, M., Tanner, H., Lehtinen, J., Levonmaa, M., Hyry, H., and Abrahamsson, P.: An Empirical Evaluation of the Process Workshop Approach. 3rd International Conference of Software Development (2005)
20. Purser, R.E. and Cabana, S.: Involve employees at every level of strategic planning. Quality progress. 30(5) (1997) 66-71
21. Riordan, C.M., Vandenberg, R.J., and Richardson, H.A.: Employee Involvement Climate and Organizational Effectiveness. Human Resource Management. 44(4) (2005) 471-488
22. Scupin, R.: The KJ Method: A Technique for Analyzing Data Derived from Japanese ethnology. Human Organization. 56(2) (1997) 233-237
23. Susman, G. and Evered, R.: An assessment of the scientific merits of action research. Administrative Science Quarterly. 23(4) (1978) 582–603
24. Vandenberg, R.J., Richardson, H.A., and Eastman, L.J.: The Impact Of High Involvement Processes on Organizational Effectiveness. Group & Organization Management. 24(3) (1999) 300-339
25. Wenger, E.: Communities of practise: learning, meaning and identity. Cambridge, UK: Cambridge University Press. (1998)

P5: Tailoring and introduction of the Rational Unified Process

Presented at EuroSPI 2007
Available in P. Abrahamsson et al. (Eds): EuroSPI 2007, LNCS 4764, pp. 7-18, 2007. © Springer-Verlag Berlin Heidelberg 2007. Reprinted with kind permission of Springer Science and Business Media.

Geir Kjetil Hanssen[1], Finn Olav Bjørnson[2] and Hans Westerheim[1]
ghanssen@sintef.no, bjornson@idi.ntnu.no, hans.westerheim@sintef.no
[1] SINTEF ICT, NO7465 Trondheim, Norway
[2] NTNT/IDI, NO7491 Trondheim, Norway

Abstract. RUP is a comprehensive software development process framework that has gained a lot of interest by the industry. One major challenge taking RUP into use is to tailor it to specific needs. This study presents a review and a systematic assembly of existing studies. We have found that tailoring RUP is a considerable challenge by itself and that tendency is turning from large complete process frameworks towards smaller and more light-weight processes.

Keywords: software development process, method tailoring, method adoption, rational unified process.

1 Introduction

As software development is a highly complex process; methodology support is a prerequisite for the completion of a successful software development project. There exist a wide variety of software development methodologies, spanning from heavy and bureaucratic processes to light-weight and dynamic processes, lately agile processes have gained a lot of interest both by the industry and academia. A more mature direction within software development methodologies is the Unified Process[1] (UP) and its commercial variant Rational Unified Process (RUP). There exist no exact figures on how many organizations that have tried and use (R)UP – in any variant; however an overview of experience reports from software engineering conferences, books and magazine publications indicate a considerable interest in UP and RUP. RUP is an extensive framework that is a collection of best practices described as a structured collection of process components; activities (what to do and how to do it), roles (by whom) and artifacts (what are the input and/or result of the activities). RUP contains detailed descriptions of these components and how they relate to each other. To establish structure, these components are organized in two dimensions; first by phases from inception to elaboration and then by a set of disciplines adhering to common SE activities. In addition, RUP is based on a few basic values; it is architecture centric, it is use-case driven and it is an iterative and incremental process. Having this completeness and complexity it is not intended to be a silver bullet process for all development project situations – RUP is a framework that must be tailored to the situation of use. It is an absolute necessity to do so to get the intentional value from using RUP.

Despite this indisputable interest, the total amount of empirical studies on the *adoption* and *introduction* of RUP is surprisingly low. A search for empirical studies identified only five studies that to some extent explain tailoring and introduction of RUP. We separate clearly between simple lessons-learned reports that don’t present information on context and study method and those that present these details as well as findings, analysis and conclusions. This leads to the aim of this paper: What do the software industry and the research community knows of the limitations, benefits, prerequisites and costs of tailoring and introducing Rational Unified Process? Thus, cost and benefit of RUP in *use* is outside the scope of this paper.

As RUP covers more or less all aspects of SE it may seem easy to take it into use. However there are many challenges in doing so successfully. How do you know which parts to keep, exclude or alter? Who should get involved in the process? How much time does it take? How is the result to be taken into use? How do you know that the result was good? To be able to answer such questions and to pinpoint further research needs, at least in part, we have done a literature review of all existing relevant studies on tailoring and introducing RUP - holding a minimum of methodological quality. In addition, we extend this compiled overview with three case studies of the introduction and use of RUP that the authors have done over the past few years [2-5] thus bringing together all available empirical experience on the topic.

This paper first describes our research method, both for the literature review and for our own case studies. Then, results are presented giving an overview of identified experience reports. A discussion summarizes findings from the literature review and own experiences giving a conclusion addressing the research aim of this paper.

2 Background: method tailoring

There exists a set of guidelines for tailoring and adoption of RUP; one book that specifically targets the issue [6] and one book that covers the issue to some detail [7]. Additionally there exists a guideline documented through a website [8]. In addition there are some guidance in the RUP documentation itself [4] or RUP-related books, however these guidelines tends to be superficial. Despite the existence of these guidelines the authors have not been able to find any experience reports evaluating their outcome and suitability. On the other hand, there exist a set of experience reports addressing tailoring and adoption of RUP done in other ways. These experience reports are summarized and analyzed later in this paper.

The term methodology is defined as "A body of methods, rules, and postulates employed by a discipline: a particular procedure or set of procedures" by the Merriam-Webster dictionary [9]. Basically, a methodology describes how someone, e.g. an organization performs a task, e.g. software development. In our context we talk about methodologies for running projects with a defined customer having more or less defined goals initially.

The process of adapting RUP can possibly take many forms. IBM Rational, the provider of RUP has defined the Process Engineering Process (PEP) [8]. This is a comprehensive adaptation process requiring a fairly big amount of resources (people and time). This may very well be appropriate for larger companies, but for the small ones this process may be too expensive. Adaptation of a framework, such as RUP, can take one of (at least) three approaches. The first is to do it in one step, for each project, thus representing a heavy job in each case. This can be justified for large projects. This approach may be called situational method engineering, as defined by ter Hoefstede and Verhoef in [10]. The second approach is to do an up-front adaptation producing a subset of the framework, still being a framework, but now tuned to the organizations general characteristics (technology, customers, domain, traditions etc.). This is the intentional process of PEP and may be called method engineering, as defined by Brinkkemper in [11]. The thirds approach is to first identify and describe a set of recurring project types. Having knowledge of characteristics and differences of these types, an adaptation is done for each type. No matter which approach being used; in the last step, a final adaptation is done to each case (project).

Adapting RUP in practice means to decide on which process elements to keep, remove, alter, add or merge. These decisions can be based on assumptions, experience, goals and visions. It is the quality of this underlying knowledge and experience that determines how good these decisions are. Having decided the content and principles of a process it must be made available to the users – the project team(s). Traditionally process descriptions have taken the form of voluminous printed descriptions. Today the most common form is through web-based process guides, RUP Online is such an example. In the case of RUP, IBM Rational provide a set of software tools to assist the reengineering of the process elements of RUP to build a coherent web based presentation of the result. Edwards et al. [12] emphasize the importance of actively involving stakeholders in the process of tailoring situational specific methods. This will both ensure that necessary detailed information becomes available and affects the tailoring process and that the resulting process actually is taken into use due to ownership and relevance. Various acceptance models such as TAM, TAM2, PCI and others [13] may help to explain and underline the importance of involving stakeholders that, after the tailoring, are going to use or be affected by the resulting process. For example, stakeholder participation may affect the *Usefulness*-construct (the extent to which the person thinks using the system will enhance his or her job performance) and the *Ease-of-use*-construct (the extent to which the person perceives using the system will be free of effort).

3 Method

In this chapter we first describe the study methods used in our own three studies – each description is based on four parts: 1) a brief overview of the study context, 2) study aim, 3) data collection procedures and 4) method for data analysis and finally, in the last part of the chapter we present the method used to perform the literature review.

Case study A:

Context: Company A is a Norwegian software consultancy company with 50 employees mainly developing software systems with heavy back-end logic and often with a web front-end, typically portals. However, they also develop lighter solutions with most emphasis on the front-end. All development is done in the form of projects. The authors have followed A for a period of five years - having a varying focus over these years; First we studied how A initially used RUP, out-of-the-box, with no restrictions or guidelines. The study is reported in [3]. Secondly, we carried out an action research project to follow A in an attempt to tailor RUP to a predefined project type. The study is reported in [2]. Thirdly, and finally, we have carried out a case study of a pilot project at A using a heavily downscaled variant of RUP documented in the form of an internal Wiki-web. The results from this study are still not published, however reported in this article.

Study aim: For the three studies, the study aims were respectively; *to present an industry case to provide lessons learned and answers with respect to process uptake and effect*. The second study aimed *to provide others considering remodeling and adapting a process framework in general, and RUP particularly, an insight in how this has been done in a small software company.* The third study aimed *to study the use and effects of an extensively downscaled variant of RUP documented in the form of a Wiki-web*.

Data collection: For the first study we first interviewed four project managers (claiming to be using RUP in four projects) to make a usage map per project to see what parts of RUP actually was being used. Then, we arranged semi structured interviews with five employees with varying roles to document main experiences and find potential explanations for use/no-use of RUP. For the second study we took an action research approach [14] following A in the whole process of tailoring RUP, as a group-process, to a defined project-type. In the third study we have interviewed the project manager and analyzed internal mid term- and end- PMA-evaluations [15] of the pilot project being studied.

Analysis: As all three studies have been descriptive with no hypothesis to validate we have done a qualitative analysis. For the first study, interviews were documented on-the-fly in a usage-map (excel spreadsheet) showing which RUP process components had been used or not with potential explanations from the interviewees. Further on, the interviews were transcribed and analyzed using the constant comparison technique [16]. In the second study which was organized according to the principles of action research our report [2] contains a discussion that extracts and summarizes key learning's. In the third study we also used the constant comparison technique to extract key learning's from the transcribed interview and the internal project evaluations.

Case study B:

Context: Company B is the software development department (300 persons) within a large Norwegian company with a total of 2000 employees. B is focused at both software development and consulting services within the domain of banking and transportation services. The authors have followed B over period of two years, entering the scene about a year after the company's RUP specialization had been taken into use by projects. This study is reported in[4].

Study aim: The aim of the study was *to investigate the level of use of a large-scale RUP specialization, explaining positive and negative experiences using the tailored process and reasons for use/no-use.*
Data collection: In this case study we used three main sources of information; 1) a main contact person which was the leader of the tailoring of RUP prior to our study, 2) the process advisory board responsible of the tailoring and the introduction of the new process in the organization and 3) project managers and software developers. Our main method of data collection was workshops and semi structured interviews with these roles. We had three workshops with the project advisory board; information was recorded on-the-fly using mind-maps. We did two rounds of interviews, the first – interviewing representatives from eight projects face-to-face, mainly project managers. The second round of interviews was carried out one year later with the same eight interviewees, this time over telephone. All 16 interviews were recorded and transcribed for later analysis. The aim of the interviews was to document experiences from the introduction of the tailored RUP, find effects – both positive and negative, and to investigate the level of use and correspondingly explanations.

Analysis: All transcribed interviews was analyzed using the constant comparison technique, the first eight interviews were coded and analyzed using the NVivo™-tool, the last eight were coded manually by two researchers in pair using a whiteboard. Lessons learned and experiences were counted across the interviews to find key learning's of most significance.

Case study C:

Context: Company C was a company specializing in the development of web applications with a high emphasis on the user experience of the web sites. The company had software developers and psychologists employed. The latter ones worked as producers, specifying the look and feel of the web sites, as well as the logical aspects of the use of the web pages. The company did develop both ecommerce applications and more entertainment types of sites. This study is reported in [5] and [17].

Study aim: The aim of the study was to investigate how RUP could support the specifications and development of non-functional parts of a web site. The company had its own tailored RUP, where the original disciplines and the structure of RUP were not changed. The tailoring was a new user experience discipline, with dedicated activities to be performed by new roles.

Data collection: In this case study the main data source was the conducted Postmortem [15] analyses. Data from six different projects is included in the case study. The tailoring of RUP was already in place when the researchers started to cooperate with the company.

Analysis: The data in the PMA reports was analyzed using constant comparison.

Literature review method:

A systematic review is a strategy for gathering and systematizing results from several independent studies sharing more or less the same thematic focus. The intention is to establish a compiled overview of all relevant experiences and to identify gaps in existing knowledge, thus implicating the directions for further research. In this case we did a simplified review inspired by the guidelines described by Kitchenham [18], hence we call it a litterature review.

Systematic reviews have traditionally been used to systematize quantitative research, typically statistical meta-analysis. However, most software engineering method-focused experience reports so far are qualitative single-case studies. We therefore needed to adopt practices to be able to systematize qualitative data. This resulted in a review-protocol that we used to 1) define a common research question, 2) search for relevant literature, 3) select studies to include in an analysis and 4) systematize findings and lessons learned.

Step 1 - A common research question: We defined the following question for the review: *What are the challenges, prerequisites and success criteria's for tailoring, introducing and using a software development method, e.g. RUP?*

Step 2 - Finding relevant literature: The following SE index databases; ISI Web of science, Compendex and ACM Digital Library were searched using the phrase *unified process AND software.*

Step 3 – Select studies to keep: All three authors participated in the evaluation of the search results using the following routine:

- **Deselect on title:** a coarse deselection of studies was done based on title, removing studies with an obvious wrong focus. The exclusions and inclusions were based on a few simple selection criteria's: The study aim or topic had to be within the frames of *tailoring/adopting/specializing/introducing the Unified Process or Rational Unified Process* This resulted in 100 unique studies.
- **Deselect on title and abstract**: The second selection criterion was: the study must present empirical data. This left 36 studies.
- **Deselect on full text**: Studies was excluded if they had insufficient quality with respect to 1) a well defined and limited study aim, 2) an adequate description of the study method, 3) a sufficient description of the study context, 4) a presentation of the study results, 5) a thorough analysis of the results and 6) giving conclusions or answers with respect to the defined study aim. This left 5 studies.
- **Final, group based selection**: Each resulting study was reviewed by each of the three authors discussing the six quality criterions defined above. This final step left 2 studies.

Step 4 - Systematize findings and lessons learned: The main learnings or conclusions from the resulting studies were identified and expressed as claims. A claim can be seen as a hypothesis supported by at least one study.

4 Results

Case study A:

The first part of the study, addressing RUP-use out-of-the-box concludes that a direct use of a framework, such as RUP, with no assistance, tailoring or guidelines results in low use. Introducing a methodology such as RUP is an investment beyond the license fee. In this case the outcome could have been better if the introduction of RUP was carefully managed and not left as an autonomous effort in each project. The second part of the study concludes that a success factor in tailoring RUP to a defined project type is to have focus on the features of the defined process and that a tailoring workshop should consist of persons with proper experience from case projects of the defined type. In the third study we saw that the main objection with the use of the small footprint process guide was lack of content, the project manager typically had a demand for more and better check lists. However, the content was still under development. The project manager commented that it has to be a balance between content size and the lightness as one of the main positive experiences was the simplicity of the guide – it was easy to find relevant guidance. As the process guide is a Wiki-web the project manager clearly saw a need of defining an editor role as editing is free to all and may compromise the content. The content which basically is a collection of activity descriptions organized over the four RUP phases seemed appropriate for the case project, only four new activity descriptions was suggested. Beyond task guidance the project manager strongly demanded practical process support tools such as estimation models, project follow-up support, a testing framework etc. When asked to comment the difference between this light process guide and the complete RUP the project manager emphasized the ease of use and clear relevance of the new guide as opposed to RUP's well of information that may be hard to find one's way through. However, interestingly, a definite premise of using such a minimum version of RUP is that the user must have an good understanding of the principles of RUP.

Claim A.1: RUP, out-of-the-box is over-comprehensive and will provide more confusion than guidance and consequently low uptake and use.
Claim A.2: Tailoring RUP efficiently must be based on best practice from the native organization and relevant project cases.
Claim A.3: RUP may be downscaled extensively to increase relevance and ease of use, however, a successful use requires a good knowledge of RUP principles.

Case study B:

The findings resemble with known models of technology acceptance[13]; little knowledge of RUP and thereby low motivation results in low or no use. On the other hand, knowledge and motivation for RUP results in medium/extensive use. In relation, education seems to be an important factor, not only prior to the process but also continuously trough the use. Further on, we found that management support seemed to be an important factor with respect to uptake and to continuously improve the process during use; this also resembles with other similar studies[19].

Claim B.1: Low knowledge of RUP creates low motivation and further low uptake and use.

Claim B.2: Management support is a success factor in tailoring and using RUP efficiently.

Case study C:

The main result, when it comes to introduction of RUP, is that formalization of roles makes them more visible and understandable to others in a project. In this case, new roles related to graphical design were added to the RUP process resulting in a higher acceptance from more technical roles which consequently increased the uptake and use of RUP in the project.

Claim C.1: Explicit definition of roles makes them visible to other project members and thus positively affects the use of the process.

Our search for empirically justified claims on RUP tailoring and adaptation resulted in only two study reports; a clear signal that more research is needed in this area. In this chapter we summarize the claims these papers add to the research community. To assess the validity of these claims, we also include a short summary of the setting and research method described in each of the papers. The papers we identified were by Folkestad et.al. [20] and Bygstad [21].

Folkestad et.al. [20]:

Context: The specific case being studied was a project to transfer an existing system from mainframe architecture to a client-server based architecture. The company saw the project as an opportunity to rebuild and enhance the competence of their staff and was willing to spend resources on this. They chose to use a version of Unified Process as their software development approach. The size of the project was about 30 man-years and lasted three years.

Study aim: The study aims are clearly stated as 1) Identify the effects of changing to a new process. 2) Identify the causes for these changes. 3) Identify what properties of the new work process that was instrumental in the change.

Data collection: The data was gathered after the project had been running for one year. The main sources were seven semi-structured depth interviews with members of the software developer group. In addition some data was gathered through informal discussions and from the business' documents regarding the development process and the project.

Analysis: The data was analyzed qualitatively using a method called Activity Theory, which can be considered "a framework for the understanding of human activity".

Limitations: Openly discussed in the paper. Since it is a single case study, it is not easy to generalize the results. Factors like openness, flat hierarchy, and confident staff may be the cause behind the results, just as much as UP itself.
Findings: We have extracted the following findings based on this paper:

Claim R.1: The iterative approach of Unified Process will ensure large effects in terms of learning.
Claim R.2: Unified Process will improve on communication and work distribution in a company.
Claim R.3: Unified Process helps constrain activities and leads to developers being more focused on their tasks, and hence it has a positive influence on productivity and quality.
Claim R.4: As a project develops, elements of Unified Process will become internalized and become tools for the developers. Or in other words, the developers will focus less and less on UP in itself, but focus more on following the practices that they decide to adopt.

Bygstad : [21]

Context: A RUP development project at Scandinavian Airline System (SAS), carried out by the Scandinavian IT Group (SIG) (owned by SAS). The goal of the project was to establish a web based marketing channel, enable easy publishing and integrating it with the existing booking systems. SAS had chosen RUP as their standard software methodology two years prior to this project. RUP was tailored to the project, and was linked to established practices in SIG.

Study aim: The research questions are 1) how can the project manager control the integration challenge? And 2) what support is there in the software engineering frameworks, like RUP?

Data collection: The case was followed for 18 months. Interviews were conducted over three intervals, project meetings were observed and project documentation analyzed.

Analysis: All data was coded with in-vivo codes, using only domain (project) terms. Then each iteration of the project was analyzed qualitatively using constant comparison methods.

Limitations: There is no discussion concerning external validity, but since it is a single case study, the results may not be easy to generalize. The internal validity is discussed in the paper with emphasis on how they addressed the principles of dialogical reasoning, multiple interpretations and member verification in their analysis.

Findings:
Claim R.5: RUP provides good support for internal technical integration and poor support for external technical integration.
Claim R.6: RUP provides weak support for internal stakeholder integration throughout a project.
Claim R.7: RUP provides strong support for external stakeholder integration in the early phases, but weak support in the later phases.
Claim R.8: RUP gives strong declarational support to step-wise external integration, but too little practical support.

Claim R.11: Using RUP as a basis, linking it to existing best practices results in a process that is actually used.

5 Discussion

The search for relevant empirical studies, with sufficient quality, on tailoring and introduction of RUP resulted in only two study reports. In addition to our three own studies this forms a very small experience base and it has shown to be hard to see any trends across these studies.

From the studies we see that RUP initially is too complex to be used without any tailoring which in practice means that the project manager must make more or less ad-hoc decisions. This becomes an error prone process if the knowledge of the content of RUP is low and thus makes it hard to decide upon which elements to keep, alter or avoid [3]. The RUP-online documentation is a comprehensive collection of process elements and their relations containing about 3700 web pages – which makes it necessary to have a detailed knowledge about the content to be able to select a consistent subset suitable for a given context of use. In the first attempt to deselect RUP elements in case study A we saw that insufficient knowledge of such details quickly became a problem. In case study B a dedicated team needed to get assistance from a trained RUP mentor to be able to accomplish a successful tailoring. In the second attempt in case study A, a bottom-up approach was used – building a small process guide based on existing best practices using RUP merely as inspiration rather than a commodity. This approach made it at least possible to accomplish the task and resulted in a complete process guide that was taken into use by project teams. In this case, almost all users of this heavily downscaled RUP-process had very high knowledge of RUP through training. This made it possible to use simplistic guidelines as the users knew the details or at least where to find them when needed. The resulting process guide itself in case A was a simple overview of the most important high-level tasks to perform in a development project – no templates or process maps were included. So, the resulting process and its web-based representation can be characterized as minimalistic, thus rising the question what RUP is; how much do you have to keep unaltered to still call it RUP and when is it merely inspired by RUP that by it self is a collection of already existing best practices and guidelines? As a contrast to case A where the basic knowledge of RUP was high we saw in case B that the intended users had little knowledge which clearly affected their motivation for use which consequently also resulted in low uptake of the new process - even though it in this case was tailored to their project characteristics by a dedicated tailoring team. Other studies also support this in the case of acceptance and uptake of electronic process guides [19]. It is reasonable to believe that low knowledge negatively affects these motivational factors. Further on, in case B, we found that management support was a success factor – one project in this case study was found to actually use RUP and report a certain level of success of doing so. In this case the management had been clear in their expectations that the project should use RUP and supported this. In other projects in the same case study, management was more absent which made the project members use their own varying best practices, thus hampering the goal of establishing a corporate unified development process. Another potential success factor for uptake was found in case study C. As RUP clearly defines roles it became evident how each role was needed and how they related to each other through joint activities and shared artefacts. This increased the acceptance of existing roles that was not documented to be a part of the total development process. We have not followed our own cases to assess the use of RUP over time, however Folkestad et al. found that developers, over time, will focus less and less on the process in itself, but focus more on following the practices that they decide to adopt [20]. Thus, the value of introducing RUP may have important effects when it comes to learning a new shared process.

An interesting note in the context of RUP and the challenge of making it fit to local needs and context is the recent spirited development of agile processes [22]. Ivar Jacobson, one of the contributors to RUP has recently initiated a total remake of RUP, resulting in something called the Essential Unified Process (EssUP). This is intended to be a great improvement of RUP and Jacobson says in a whitepaper [23]: "*The Unified Process became too heavy, the process improvement programs required too much boring work…*". This is interesting since RUP for years has been marketed as a framework that could help most software organizations in professionalizing software development effectively. EssUP can simply be described as a combination of RUP – which may be seen as a heavy type of process – and agile software development principles [24]. Our findings, both from our own studies and others support this view that RUP is too heavy and that it may require too much tedious and difficult work. The question is; will a join of RUP and agile be a better approach? Others as well has addressed the challenge of making RUP simpler and agile which, in sum, can be seen as a shared opinion that RUP has its limitations despite its comprehensiveness. This adds to our findings summarized in this paper.

RUP has since its creation gone through several transformations, all leading towards a more agile approach of designing and developing software. This has resulted in various variants and spin-offs of the process, followed by numerous books and even more presentations, speeches, courses and consultant services. It is hard to predict where this will end; however, based on our findings we see a clear need of simplifying RUP (and other processes) to ensure uptake and efficient use. The development turns clearly towards the agile side of the spectrum – perhaps in search for a balance between discipline and agility [25].

6 Conclusions

Based on our own, and a few other empirical studies on tailoring and introduction of RUP into development organizations we found that there exist few or none (reported) direct success stories. All experiences pull in the same direction; RUP is, out of the box, too complex, however, tailoring it to specific needs is also too complex. Looking at the evolution of RUP itself over the past years and the cases we summarize here we see a clear need for, and movement towards, a more agile process that can bee tailored with less effort.

Acknowledgments

The authors would like to thank the participants from the case companies. We would also like to thank the SPIKE and EVISOFT projects, funded by the Research Council of Norway under grant 156701/220 and 174390/I40.

References

1. Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Object Technology Series, ed. G. Booch, I. Jacobson, and J. Rumbaugh. 1999, Reading, Massachusetts: Addison Wesley Longman Inc. 463.
2. Hanssen, G.K., Westerheim, H., Bjørnson, F. O. *Tailoring RUP to a defined project type: A case study*. in *PROFES*. 2005. Oulo.
3. Hanssen, G.K., Westerheim, H., Bjørnson, F. O. *Using Rational Unified Process in an SME - A Case Study*. in *EuroSPI*. 2005. Budapest.
4. Westerheim, H., Hanssen, G. K. *The Introduction and Use of a Tailored Unified Process - A Case Study*. in *Euromicro*. 2005. Porto, Portugal.
5. Westerheim, H. and G.K. Hanssen, *Extending the Rational Unified Process with a User Experience Discipline: a Case Study*, in *EuroSPI*. 2006: Joensuu, Finland.
6. Bergström, S., Råberg, L., *Adopting the Rational Unified Process*. 2004, Addison-Wesley. p. 165-182.
7. Kroll, P. and P. Kruchten, *The Rational Unified Process Made Easy - A Practitionare's Guide to the RUP*, ed. O.T. Series. 2003: Addison Wesley.
8. *Rational PEP*. Available from: http://www-1.ibm.com/support/docview.wss?uid=swg21158199.
9. *Merriam-Webster dictionary*.
10. ter Hofstede, A.H.M. and T.F. Verhoef, *On the feasibility of situational method engineering.* Information Systems Journal, 1997. **22**(6): p. 401-422.
11. Brinkkemper, S., *Method engineering: Engineering of information systems development methods and tools.* Information and Software Technology, 1996. **38**(4): p. 275-280.
12. Edwards, H.M., J. Barrie Thompson, and C.J. Hardy. *Developing situationally specific methods through stakeholder collaboration*. in *Computer Software and Applications Conference (COMPSAC)*. 1998.
13. Riemenschneider, C.K., B.C. Hardgrave, and F.D. Davis, *Explaining Software Developer Acceptance of methodologies: a Comparison of Five Theoretical Models.* IEEE Transactions on Software Engineering, 2002. **28**(12): p. 1135 (10).
14. Avison, D., et al., *Action Research.* Communications of the ACM, 1999. **42**(1): p. 94 (4).
15. Birk, A., T. Dingsøyr, and T. Stålhane, *Postmortem: Never Leave a Project without It.* IEEE Software, 2002. **19**(3): p. 43 - 45.
16. Seaman, C.B., *Qualitative methods in empirical studies in software engineering.* IEEE Transactions on Software Engineering, 1999. **25**(4): p. 557-572.
17. Westerheim, H., T. Dingsøyr, and G.K. Hanssen, *Studying the User Experience Discipline extension of the Rational Unified Process and its effects on Usability - The design of a case study*, in *Empirical Studies in Software Engineering:*

Proceedings from the first international workshop, december 2002, C. Bunse and A. Jedlitschka, Editors. 2002, Fraunhofer IRB Verlag: Rovaniemi, Finland. p. 69 - 74.

18. Kitchenham, B., *Procedures for Performing Systematic Reviews*. 2004, Keele University and Empirical Software Engineering National ICT Australia Ltd. p. 33.
19. Dybå, T., N.B. Moe, and E.M. Mikkelsen. *An Empirical Investigation on Factors Affecting Software Developer Acceptance and Utilization of Electronic Process Guides*. in *METRICS*. 2004. Chicago, USA.
20. Folkestad, H., E. Pilskog, and B. Tessem, *Effects of Software Process in Organization Development – A Case Study*, in *Learning Software Organizations (LSO)*. 2004.
21. Bygstad, B., *Controlling Iterative Software Development Projects: The Challenge of Stakeholder and Technical Integration*, in *Hawaii International Conference on System Sciences*. 2004: Hawaii, USA.
22. Cockburn, A., *Agile Software Development*. The Agile Software Development Series, ed. H.J. Cockburn A. 2002: Addison-Wesley.
23. Jacobson, I., P.W. Ng, and I. Spence, *The Essential Unified Process – a Fresh New Start*. 2006.
24. *Agile Manifesto*: http://www.agilemanifesto.org/.
25. Boehm, B. and R. Turner, *Balancing Agility and Discipline - A Guide for the Perplexed*. 2004: Addison-Wesley. 266.

P6: Improving the Effectiveness of Root Cause Analysis in a Retrospective Method: a Controlled Experiment

Submitted Journal of Information and Software Technology, August 2007.

Finn Olav Bjørnson[1], Alf Inge Wang[1] and Erik Arisholm[2,3]

[1]Norwegian University of Science and Technology Department of Computer and Information Science, Sem Sælandsvei 7-9, 7491 Trondheim, Norway,
ph:+ 47 73 59 87 16, fax: + 47 73 59 44 66,
e-mail: (Finn.Olav.Bjornson, Alf.Inge.Wang)@idi.ntnu.no

[2]Simula Research Laboratory, Department of Software Engineering,
P.O.Box 134, 1325 Lysaker, Norway, ph: +47 67 82 82 00, fax: +47 67 82 82 01,
e-mail: erika@simula.no

[3]Dept. of Informatics, Univ. of Oslo, PO Box 1080 Blindern, N-0316 Oslo, Norway

Abstract

Retrospective analysis is a way to share knowledge following the completion of a project or major milestone. However, in the busy workday of a software project, there is rarely time for such reviews and there is a need for effective methods that will yield good results quickly without the need for external consultants or experts. Building on an existing method for retrospective analysis and theories of group involvement, we propose improvements to the root cause analysis phase of a lightweight retrospective analysis method known as Post Mortem Analysis (PMA). In particular, to facilitate brainstorming during the root cause analysis phase of the PMA, we propose certain processual changes to facilitate more active individual participation and the use of less rigidly structured diagrams. We conducted a controlled experiment to compare this new variation of the method with the existing one, and conclude that in our setting of small software teams with no access to an experienced facilitator, the new variation is more effective when it comes to identifying possible root causes of problems and successes. The modified method also produced more specific starting points for improving the software development process.

Keywords: Retrospective Method, Software Process Improvement, Controlled Experiment, Knowledge Management, Post Mortem Analysis.

1 Introduction

In today's software engineering industry, it is critical to improve software development processes. In this context, one lesson that may be learned from general efforts to improve processes, such as total quality management and standardisation, is that the ability to learn from past success and failure is a central factor for success [8]. Learning from the past involves knowledge management, or creating a ”learning software organisation”, which is defined by Dybå [11] as *”A software organisation that promotes improved actions through better knowledge and understanding*”.

Keegan and Turner [16] claim that, in general, software development is conducted at too fast a pace. In 2001, they performed a study on project-based learning practices in 19 European software development companies. They found that project team members frequently did not have time for meetings to review lessons learned. Where recommended process models did exist, these were seldom used. In an editorial in IEEE software in 2002, Glass [14] stated that the software engineering field is so busy that there is rarely time to think of how development could go better, not just faster. He further claimed that companies should pause from time to time to learn the lessons they had been through. He recommended reviewing performances on completed projects (project retrospectives) as a good way of learning.

There is a principle in agile software development that states that “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.” [1]. In accordance with this principle, iterative and light retrospective sessions have been suggested for use in agile projects [4, 9, 18]. Myllaho *et al*. state that the small teams and short iterations of extreme programming will affect how retrospective workshops can be conducted [20]: *“The workshops needs to be short and effective, i.e., not taking too much effort from the project team, yet yielding immediate and visible outcomes to motivate the project team for further such activity.”*

In this paper, we take as our starting point an existing, lightweight retrospective method known as the Post Mortem Analysis (PMA) [2]. We propose a modified method that exploits theories on brainstorming and group performance combined with the notation of causal maps. The effectiveness of the original PMA and our revised PMA is compared in a controlled experiment, using a quantitative measure. We also assess qualitative differences in the results of the two approaches. The main research questions we address are these:

1) *Is the revised PMA method more effective than the old PMA method?*
2) *How do the two methods differ in their result?*

The remainder of this paper is structured as follows. Section 2 discusses related work on retrospectives in software engineering. Section 3 presents the two lightweight methods that were used in the experiment. Section 4 describes the design of the experiment. In Sections 5 and 6, quantitative and qualitative results, respectively, are presented. Section 7 contains a discussion of the results. Section 8 concludes and suggests avenues for further research.

2 Related Work

According to Rising *et al.* [21], retrospective analysis as a method for learning from work experience was identified in 1988 by Joseph Juran and named ”Santayana review” in homage to the philosopher George Santayana. Since then, many organisations have used many variations of the method and under many different names. We adopt Dingsøyr's definition [8], such that retrospective analysis is a *"collective learning activity, which can be organised for projects either when they end a phase or are terminated. The main motivation is to reflect on what happened in a project in order to improve future practice - for the individuals that have participated in the project and for the organisation as a whole."* Dingsøyr lists the most common names for retrospective analysis in [8]: "project retrospectives", "post mortem analysis", "postproject review", "project analysis review", "quality improvement review", "autopsy review", "after action review", and "touch down meetings". For the remainder of this paper, we use the term 'retrospective analysis' to denote the corpus of these methods and the term 'Post Mortem Analysis' (PMA) to refer to the specific method we investigated.

Myllyaho *et al.* [20] conducted an extensive literature review within the software engineering and management literature, with the aim of reviewing retrospective analysis as a project-based learning technique. The results suggest that the use of retrospective analysis is well worth the effort, and that a simplified or 'lightweight' version of PMA can be beneficial when time is a factor.

Dingsøyr [8] discusses the importance of retrospective analysis as a method for sharing knowledge in software projects and gives an overview of the methods of retrospective analysis that are employed in the field of software engineering. In particular, Dingsøyr presents three lightweight methods of retrospective analysis, which are presented in Whitten [25], Collison and Parcell [6], and Birk *et al.* [2]. To give an overview of key differences in retrospectives, we present his comparison of the three methods (
Table 10).

Table 10: Summary of selected differences among three methods for conducting retrospective analysis, taken from [8]

	Whitten	**Collison and Parcell**	**Birk et al.**
Whom to invite?	From each major participating organisation	All project members, possibly new project	All project members
Homework?	Yes	No	No
Type of discussion	Open	Open	Structured
Output	Recommendations	Guidelines, Histories, Names of People, Key artefacts	Structured report on issues that went well and those that could be improved

Desouza *et al.* [7] compared two kinds of output from retrospective analysis: traditional reports and stories. The comparison can be found in Table 11. They also identified four factors that should affect the choice of writing the result of the PMA as a report or as a story: (1) the nature of the project, (2) the cost you are willing to bear, (3) how much organizational impact is desired, and (4) what lessons you wish to convey.

Table 11: Reports versus stories, taken from [7]

	Reports	**Stories**
Structure of Knowledge	Highly structured	Semistructured

Cost to prepare	Low	High
Richness of Knowledge	Low	High
Ease of comprehension	Easy	Medium
Ease of recall	Difficult-Medium	Easy

Stålhane *et al.* [23] conducted an assessment of two retrospective methods. One was based on the PMA of Birk *et al.* and the other consisted of structured interviews. The main focus of their research was to determine whether there are situations in which one method performs better than the other. They found that this depends on whether a focused or broad analysis is desired. For a focused analysis, the semi-structured interviews worked better than the PMA. For a broad analysis, the PMA worked better and yielded more surprises.

3 The PMA Methods Used

In this section, we describe the methods that we adapted for the PMA. The original method we used was the one suggested by Birk *et al.* [2, 10, 12, 17, 23] (see Table 10) with structured reports as output (see Table 11). In what follows, both the original and the modified method are described in detail.

3.1 PMA Method 1: The Original

The aim of this method is to bring together project participants and have them discuss what went well and what could be improved, and to analyse the root causes. Birk *et al.* use two techniques to carry out the PMA. To discover the positive and negative experiences, they use a focused brainstorm method called the KJ-method [22], resulting in affinity diagrams. To analyse the causes of these experiences, they performed root cause analysis using fishbone diagrams (also known as Ishikawa diagrams, in reference to their inventor Dr. Kaoru Ishikawa, a Japanese quality control statistician).

The postmortem meeting itself had the following four steps:

1. Introduce the PMA method and explain the purpose of the review.
2. KJ-session 1: Elicit positive experience
3. KJ-session 2: Elicit negative experience
4. Perform root cause analyses using fishbone diagrams for the most important positive and negative experiences.

3.1.1 The KJ-sessions

KJ-sessions are conducted as follows. Each participant receives a number of post-it notes and is asked to write down what they regard as the most significant experiences from the project. After everyone has finished writing, each participant puts a note on a whiteboard while explaining what he means by it. The process is repeated until all the notes have been presented, as illustrated in Figure 6a). Once all the notes have been placed on the whiteboard, the whole group discusses them and groups them according to similarity in concept. Each group of notes is then given a name, as illustrated in Figure 6b). Possible connections between groups can be marked with arrows if required. In our study, each participant received five post-it notes and the entire process was repeated twice; first for positive experiences (KJ-session 1), then for negative experiences (KJ-session 2).

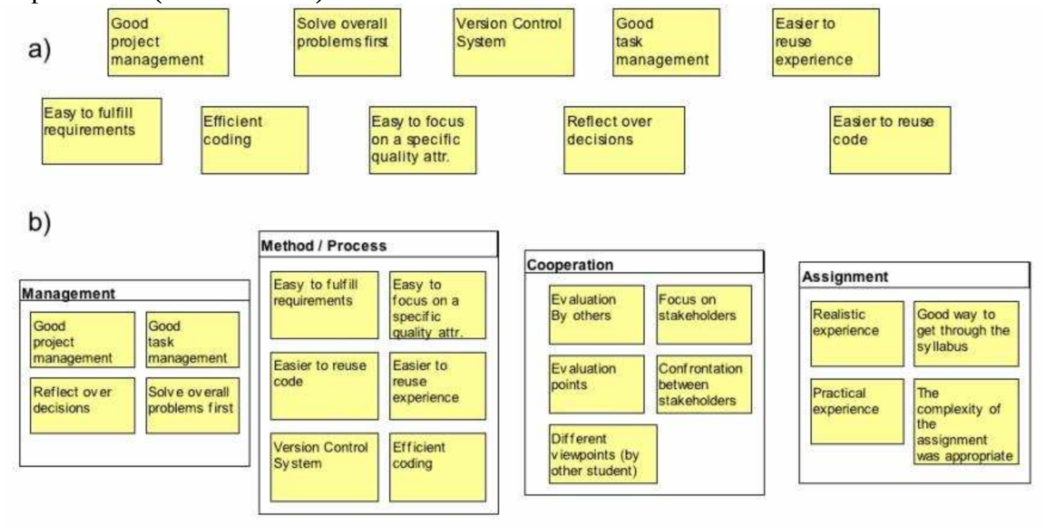
a)
Good project management
Solve overall problems first
Version Control System
Good task management
Easier to reuse experience
Easy to fulfill requirements
Efficient coding
Easy to focus on a specific quality attr.
Reflect over decisions
Easier to reuse code
b)
Management
Good project management
Good task management
Reflect over decisions
Solve overall problems first
Method / Process
Easy to fulfill requirements
Easy to focus on a specific quality attr.
Easier to reuse code
Easier to reuse experience
Version Control System
Efficient coding
Cooperation
Evaluation By others
Focus on stakeholders
Evaluation points
Confrontation between stakeholders
Different viewpoints (by other student)
Assignment
Realistic experience
Good way to get through the syllabus
Practical experience
The complexity of the assignment was appropriate

Figure 6: KJ example

3.1.2 The root cause analysis method

The root cause analysis, or fishbone diagram method, needs a facilitator who takes control of the whiteboard. The group selects a (positive or negative) experience they want to analyse the cause of and the facilitator writes the name on a whiteboard and draws an arrow to it. The group then discusses what the cause of the experience might have been and as more causes are identified, the facilitator draws arrows into the large arrow, writing in the causes. If a cause has several sub-causes they are drawn as arrows into the minor arrows, as illustrated in Figure 7.

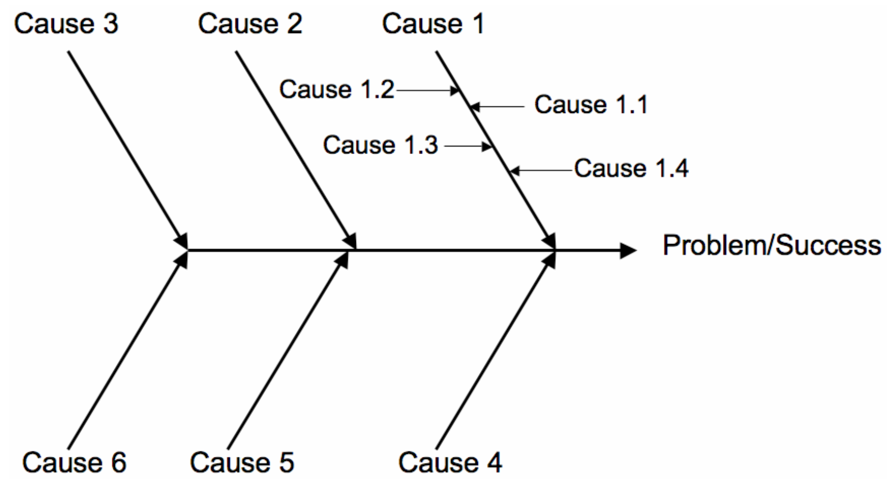

Figure 7: Fishbone example

3.2 PMA Method 2: The Revised Method

Our previous experiences of the PMA method with fishbone diagrams as a means of analysis had taught us that group activity tended to be high during the KJ phases, but that the activity seemed to dwindle as the groups proceeded with the analysis with fishbone diagrams. This tendency has also been observed by Stålhane *et al.* [23]. We wanted to increase the level of participation during the analysis phase, so we examined step four in the PMA process and proposed two main changes, which were inspired by theories on brainstorming and the notation of causal maps.

3.2.1 Theory for change

The setup of the original PMA method can be seen as the group working *nominally* in the KJ session and *interactively* in the root-cause analysis phase. A group is defined as nominal if its members work independently, but in each other's presence. A group is defined as interactive if its members generate ideas in face-to-face discussions. According to Faure [13], evidence in the field suggests that nominal groups outperform interactive groups on the number of original ideas generated in a brainstorming session. Accordingly, we attempted to make phase four more nominal. We did this by using the same technique as in the KJ sessions; namely, by using post-it notes and letting the group members come up with possible causes individually before coming together to discuss them.

In order to better accommodate the nominal brainstorm technique, we needed a more free form diagrammatic technique for presenting the results. For this, we examined the technique of causal mapping, which according to Hodgkinson [15] is one of the most popular methods for investigating individuals' cognitive representations in strategic decision making. Hodgkinson further observes that a growing number of researchers are employing one or more variants of causal mapping directly, as a means of eliciting actors' cognitions *in situ*, in an attempt to gain insights into the nature and significance of cognitive processes in organizational decision making. There exist many alternative elicitation procedures, but for our study we opted for a simple freehand mapping variety, using only the notation illustrated in Figure 3. Here, every oval represents a concept, every arrow indicates a cause-effect relationship, and the whole map represents a specific situation.

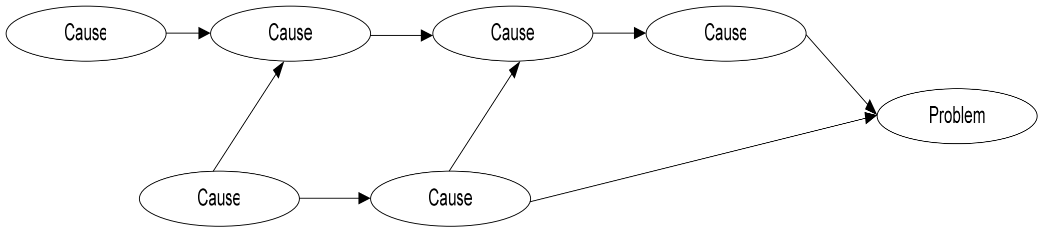

Figure 8: Causal map example

3.2.2 Practical changes

The procedure for the postmortem meeting itself is the same as in the original method outlined in Section 3.1, except for step four, for which we substituted what we call "the causal map analysis".

4. Causal map analysis: On the most important positive experience and the most important negative experience.

The new causal map analysis works as follows. All participants are given post-it notes and are asked to write down the causes of the experience to be analysed. These notes are then presented and placed on the whiteboard, much in the same way as when using the KJ-method. The group then gathers at the whiteboard and groups the causes where applicable. Cause – effect relationships are then indicated by arrows. The members are then allowed to write new notes that state deeper causes, or if causes are seen to be missing, write those in and indicate them with arrows. When the new causes have been placed on the whiteboard, the process is iterated until the group is satisfied with the analysis.

The main differences between the new and old analysis phase are:

- Forcing everyone to participate more actively by filling out the mandatory post-it notes.
- Allowing more freedom in the diagrams.

4 Research Method

This section describes the design of the controlled experiment that investigated the effectiveness of using fishbone diagrams vs. using causal maps in the root-cause analysis phase of a PMA.

We performed PMA sessions in 2004 and 2005 in which we used fishbone diagrams and causal maps, respectively. The PMA reports from the two years were analysed, and we found that participants in the sessions produced a greater number of items when using fishbone than causal maps. However, when we looked at the content of the ideas generated we found that causal maps produced a greater number of *new* items than when using fishbone diagrams. These PMA sessions were, however, not planned intentionally as a controlled experiment and we did not have control of factors that could affect the results. On the basis of our experiences from the PMA sessions in 2004 and 2005, we planned a controlled experiment and performed it in 2006. The motivation for this experiment was to limit other factors that could threaten the experimental results, such as lack of randomization of subjects, different introductions to the two PMA methods (fishbone and causal), and different working conditions and time limits for the groups.

4.1 Experimental Context

The experiment described in this paper was executed as a part of a software architecture course for Masters' students at the Norwegian University of Science and Technology. In this course, the students must carry out a software architecture project, the goal of which is to develop the software for a robot controller. The students work in groups of four to six. During the semester, the students must deliver a requirement specification, an architectural description, an architecture evaluation (using ATAM [3]) and an implementation of the robot controller according to the architecture. In the final phase of the project, the students perform a post-mortem analysis of the robot project using PMA methods as described in Section 3. The students should spend half of the time on finding and analysing positive aspects of the project and the other half on the negative aspects [24]. The number of students taking this course varies from 60 to 100.

4.2 Hypothesis Formulation

Our hypothesis was designed to assess whether the choice of analysis method (causal maps vs. fishbone) affects the percentage of new items found in the analysis phase (second phase) of a post-mortem analysis, as quantified by the dependent variable *AnalysisEffectiveness*. Thus, we wanted to investigate whether one of the post-mortem analysis methods is more effective than the other. The hypothesis was as follows:

H0: AnalysisEffectiveness(Causal maps) = AnalysisEffectiveness(Fishbone)
H1: AnalysisEffectiveness(Causal maps) > AnalysisEffectiveness(Fishbone)

The test was one-tailed, to reflect our expectation that the causal maps would be more effective than fishbone, as suggested by our previous experiences and also justified theoretically in Section 3.

4.3 Study Variables

This section defines the independent and dependent variables of the experiment and outlines how they were measured.

AnalysisMethod:
The independent variable describes whether the subjects performed the second PMA phase using fishbone diagrams or causal maps. The main differences between the fishbone diagram and the causal map approach are twofold. First, for the fishbone approach the group process is managed through one facilitator, while for the causal approach all participants will manage the process together. Second, the way the diagrams can be expressed in fishbone diagrams is different from causal maps. In fishbone diagrams, the information must be described in a strict hierarchical manner. In causal maps, there are few restrictions on how the relationships between items can be expressed.

AnalysisEffectiveness:
The dependent variable of the experiment attempts to measure the effectiveness of the PMA methods. To explain the *AnlysisEffectiveness* variable properly, we recapitulate briefly the PMA process, which consists of two main phases. In the first phase (steps 2 and 3), the participants brainstorm on either positive or negative aspects of the project and all the items found are represented as post-it notes in an affinity diagram. I_{PHASE1} is the number of items found in phase 1. In the second phase (step 4), the participants analyse one particular issue (positive or negative) to determine the reasons or background for this issue. The second phase generates a number of items, which are represented in a fishbone diagram or causal map. I_{PHASE2} is the number of items found in phase 2. To measure effectiveness, we compute how many of the items found in the second phase are new from the first phase. Thus, we can compute the AnalysisEffectiveness as

$$\text{AnalysisEffectiveness} = \frac{(I_{PHASE2} - (I_{PHASE1} \cap I_{PHASE2})) * 100}{I_{PHASE2}}$$

$I_{PHASE1} \cap I_{PHASE2}$ denotes the number of items that are common in phases 1 and 2. For example, if none of the items found in the second phase were found in the first, the effectiveness will be computed as 100%. If all of the items found in the second phase were also found in the first, the effectiveness will be computed as 0%. This means that the effectiveness will range from 0%-100%.

When counting items, two or more items that describe exactly the same issue are regarded as duplicates and are removed. Items presented in the second phase are new if no items in the first phase state the exact same meaning.

The AnalysisEffectiveness variable was measured by going through the PMA reports of the subjects. The first step was to eliminate redundancy by removing duplicate items. Two or more items were considered to be duplicates if they the exact same wording or the exact same meaning. The second step was to count items from the brainstorm phase and the items from the analysis phase. The third step was to find the items with the exact same wording or meaning from both phases, and mark these. The effectiveness was then computed by counting the number of unmarked items from the analysis phase divided by the total number of items from the same phase. To reduce the possible bias caused by subjective judgement, two researchers performed this process independently and later compared the results. In cases where there was disagreement, the items of concern were examined carefully before a decision was made.

4.4 Group Assignment

A randomized experimental design was used in the controlled experiment. Each subject (group of students) was assigned randomly to either the fishbone diagram or causal map treatment. The groups were established at the beginning of the software architecture course. A list was made available for the students to sign up for a group before a specified deadline. The students that signed this list knew each other in beforehand. After the deadline, the remaining students were assigned to groups that had open slots or to new groups. The assignment to the fishbone and causal map treatments was distributed evenly in relation to groups that were joined by students and groups that were assigned by course staff. Table 12 describes the distribution of the number of subjects (groups) to the two PMA variants. The size of the groups varied from four to six students. A total of 142 students participated in the experiment.

Table 12: Distribution of subjects in the controlled experiment

	Fishbone diagram	Causal map	Total
Number of groups	14	15	29

4.5 Experiment Tasks

The controlled experiment consisted of the following tasks:

- **Presentation of PMA method (30 min):** The two variants of the PMA method (fishbone and causal) were presented simultaneously in two different rooms by two different lecturers. The content of the presentations was analysed before they were made, to ensure that they were similar in all respects except those that pertained to describing the two variants. The first part of the presentation was exactly the same, while in the second the presentations differed in that one described the fishbone method and the other described the causal method. In the second part, the two methods were presented in a similar way and used the same number of slides. The participants asked roughly the same number of questions in each presentation, but the causal session lasted 5 minutes shorter than the fishbone session.
- **Positive brainstorm (30 min):** The participants brainstormed on positive aspects of the project and described the results in an affinity diagram. The result was recorded on a laptop PC or on paper.
- **Negative brainstorm (30 min):** The participants brainstormed on negative aspects of the project and described the results in an affinity diagram. The result was recorded on a laptop PC or on paper.
- **Positive root-cause analysis (20 min):** The issue that received the most votes from the brainstorming session on positive aspects was analysed using either a fishbone diagram or causal maps. The result was recorded on a laptop PC or on paper.
- **Negative root-cause analysis (20 min):** The issue that received the most votes from the brainstorming session on negative aspects was analysed using either a fishbone diagram or causal maps. The result was recorded on a laptop PC or on paper.
- **Write PMA-report (approx. 2 hours):** All groups involved in the PMA had to write a report on the PMA. The report had to contain (i) four diagrams and a description from the brainstorm and analysis phase and (ii) a description of their experience of doing the PMA.

4.6 Analysis

Quantitative

The purpose of the quantitative test was to determine whether or not there was a difference between the use of the fishbone and causal methods. The hypothesis was tested using a standard two-sample one-tailed t-Test assuming unequal variances. Although the t-test assumes a normal distribution, it is known to be relatively robust to mild deviations from this assumption. However, given our small sample size, it is not really possible to assess deviations from this assumption in a reliable way. Thus, to reduce potential threats to validity that might have resulted from violations of the t-test assumptions, a non-parametric Wilcoxon rank sum test was also performed. Given the small sample size, the Wilcoxon test was performed using the *Exact* option in the SAS statistical software package. The level of significance of the hypothesis test was set to $\alpha = 0.05$.

Qualitative

The purpose of running a qualitative analysis was to determine what the difference between the use of the fishbone and causal methods consisted of, if there was a difference. The qualitative analysis was performed after the results from the quantitative analysis were known.

Qualitative data were collected from three sources: (i) observation of the students by two researchers as they performed the different methods; (ii) the collection of the final reports; and (iii) a brief report that the students were told to write on their impression of the method and their experience with it. The data was analysed by hand, using a simple constant comparison method [19].

5 Quantitative Results

This section describes the quantitative results and shows the results from the hypothesis test for the controlled experiment.

5.1 Descriptive Statistics

The descriptive statistics of the experiment are shown in Table 13. The column *AnalysisEffectiveness* shows the mean value of the percentage of new items found in the analysis phase of the PMA using the fishbone and causal methods, respectively. The analysis effectiveness was 59.8% for fishbone diagrams and 78.4% for causal maps, which indicates a practically significant mean difference of 18.6%.

Table 13 Descriptive statistics of analysis method and effectiveness

AnalysisMethod	**AnalysisEffectiveness**	**Std**	**Min**	**Q1**	**Med**	**Q3**	**Max**
Fishbone	59,8%	19,8%	20,0%	50,0%	68,3%	73,3%	83,3%
Causal	78,4%	15,6%	46,2%	73,9%	80,0%	89,2%	100,0%

5.2 Formal Hypothesis Test

The two-sample, one-tailed t-Test on the difference in means resulted in a p-value of 0.0062. The corresponding exact Wilcoxon rank sum test resulted in a p-value of 0.0041. Thus, for both the parametric and non-parametric tests, the p-value is well below the 0.05 level, which suggests that there is a statistically significant difference between the analysis effectiveness of the two methods of analysis.

5.3 Effect Size

The sample's mean, data distribution, and 95% confidence interval of the mean for the dependent variable *AnalysisEffectiveness* are presented in a diamond plot, as a way to visualize the effect size of the two treatments (Figure 9). The line across each diamond represents the group mean and the vertical span of each diamond the 95% confidence interval for each group. Overlap marks are drawn below and above the means and an overlap represents a difference that is not significant at $\alpha = 0.05$. The line crossing the diagram is the entire sample mean.

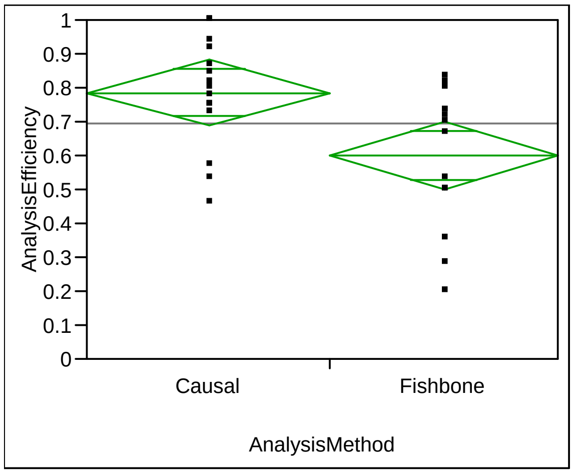

Figure 9: Diamond plot of the effect of AnalysisMethod on AnalysisEffectiveness

To further quantify the difference between the two analysis methods, we calculated a standardized effect size measure known as Cohen's *d* [5]. In our case, Cohen's *d* was calculated by dividing the difference between the mean AnalysisEffectiveness of causal maps and fishbone diagrams with the pooled standard deviation, yielding d = 1.05. Cohen suggested that if *d* is greater than 0.8, the effect size can be considered to be large.

6 Qualitative results

The quantitative tests suggest that there is a difference in effectiveness between the two methods, but what that difference consists of remained an issue. To determine what the difference consists of, we made a qualitative analysis of the final reports.

The two diagram types differ importantly in the structure that they yield. One difference concerns the number and depth of the causal links stated. The fishbone diagrams usually contained three to four main causes, and subcauses varied from none to four. The average cause effect chain was two links. By contrast, the causal maps contained from two to eight main causes and had cause effect chains up to five links long, the average being about three links. The free form of the causal maps seems to support and encourage a greater degree of analysis of causes into their relevant subcauses.

Another difference concerns the way in which causes were analysed into subcauses. The students using the fishbone diagrams would put evenly distributed subcauses on all their main causes, whether they were particularly relevant or not. The students using the causal maps would typically select a few relevant causes and create longer cause-effect chains for these, and ignore the more irrelevant main causes, such as causes outside their control.

One of the major goals of the PMA is to learn from experience and improve performance for future projects. The cause-effect chains in causal maps are longer than those in fishbone diagrams, and the causes noted seem to be more specific. It is thus easier to think of courses of action to improve performance. The longer chains yielded by the causal map approach tended to paint a more nuanced picture of the situation in the project, with general causes being stated first and more specific causes being stated deeper in the chain as the general causes are analysed.

We also observed the formation of what we called *hubs* in causal maps. Since a node can be the cause of several other nodes and also the effect of several subcauses, sometimes we observed nodes with several arrows going in and out. These nodes were very easy to identify in the diagrams and typically marked major problem spots in the projects.

However, whether the causal or fishbone method was used is not the only factor that affected the result. One observation from the qualitative analysis was that if the students chose a topic for analysis that was outside their control, the quality of the analysis was often low with regards to useful experience that could be transferred to new projects, regardless of the method they used for the cause-effect analysis.

Figure 10 and 6 show two examples from the PMA experiment that illustrate the qualitative difference we found between the resulting causal maps and the fishbone diagrams. In the causal map shown in Figure 10, the greatest number of links from a cause to the problem of focus is four. The causal map also contains hubs where one node is affected by several other nodes, e.g. the boxes ”No appointed formal project manager” and “Informal group meetings”. Such hubs usually indicate a cause that relates to many problems. Also note that many of the causes in the diagram are specific and can be addressed so that performance in the next project can be improved; by, e.g., assigning a project manager and enforcing more formal group meetings.

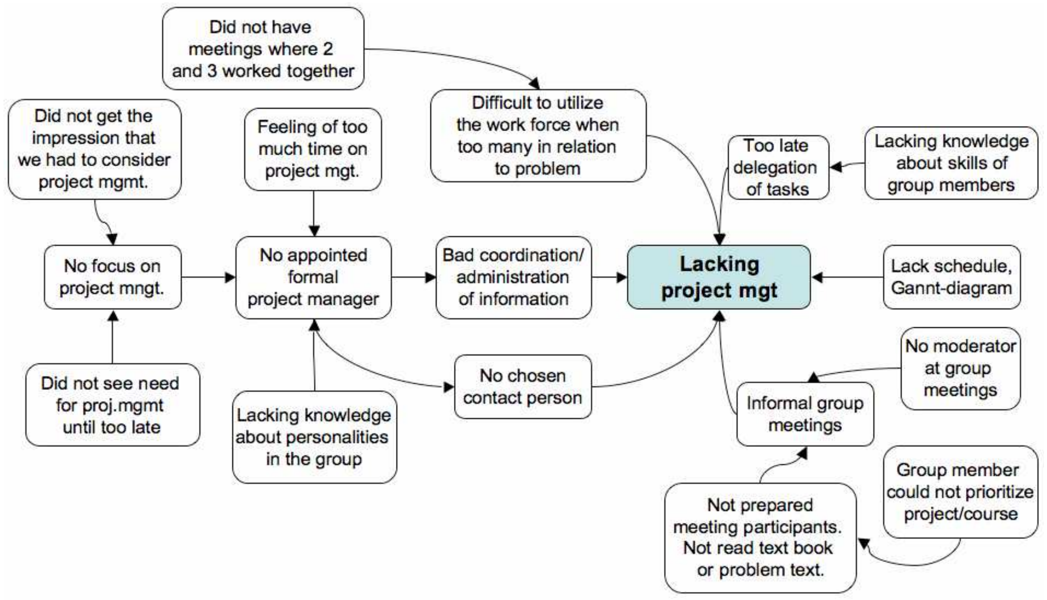
Did not have meetings where 2 and 3 worked together
Did not get the impression that we had to consider project mgmt.
Feeling of too much time on project mgt.
Difficult to utilize the work force when too many in relation to problem
Too late delegation of tasks
Lacking knowledge about skills of group members
No focus on project mngt.
No appointed formal project manager
Bad coordination/ administration of information
Lacking project mgt
Lack schedule, Gannt-diagram
No moderator at group meetings
Did not see need for proj.mgmt until too late
Lacking knowledge about personalities in the group
No chosen contact person
Informal group meetings
Group member could not prioritize project/course
Not prepared meeting participants. Not read text book or problem text.

Figure 10 Example of a causal map of a problem

Figure 11 shows a typical fishbone diagram. Compared to the causal map diagram, fewer causes are analysed into subcauses, the causes are not analysed to a depth of more than three levels, and the causes are more general.

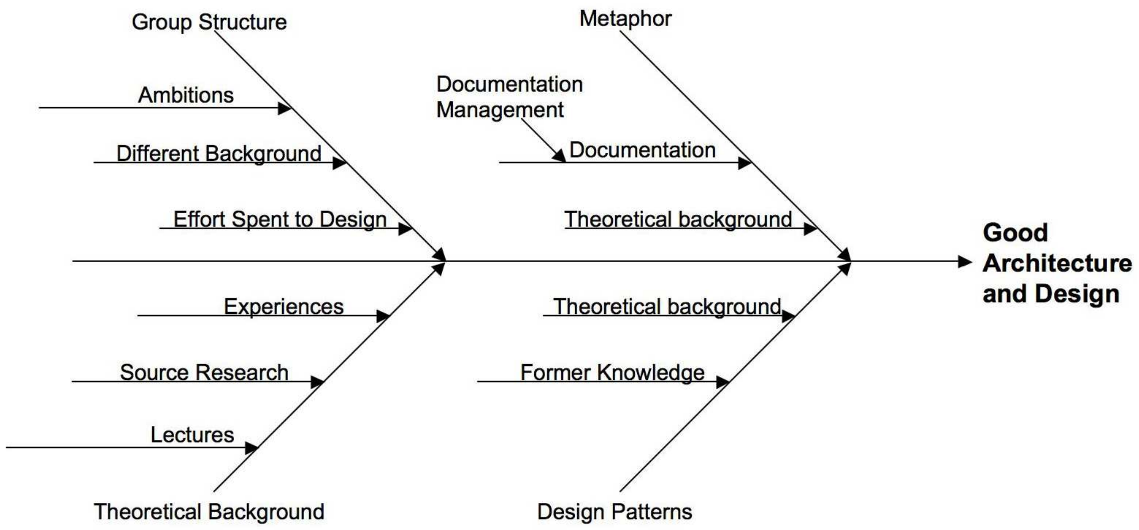

Figure 11 Example of a fishbone diagram on a success

In addition to the final reports, we observed the groups' behaviour during the PMA sessions. Our qualitative observation was that the groups using the causal map technique participated more during the analysis phase than the groups using the fishbone diagrams.

7 Discussion

In this section we discuss our findings and possible threats to the validity of our experiment.

7.1 Our results

The quantitative results presented in Section 5 showed that, in our setting with small software teams with no access to professional facilitatores, using causal maps is more effective than fishbone diagrams for analysing root causes of problems or successes in PMAs. This result can be explained by the fact that the groups that made causal maps used a nominal brainstorming technique when generating their initial ideas on causes, whereas the groups that made fishbone diagrams used an interactive technique. The observation that the nominal group technique outperformed the interactive one, is a result that is in line with earlier research on brainstorming [13].

Another possible explanation for the significant difference in effectiveness between the two approaches is that we used untrained facilitators in our PMA sessions. The difference might have been less, had we used professional facilitators to properly steer the conversations. We know that the fishbone method will benefit from an experienced facilitator who can coax the underlying causes from the participants, but there have been no tests to suggest how much the causal map method would benefit from having such a facilitator. Our observations from several PMA sessions do, however, indicate that the motivation and level of activity is generally higher when making causal maps than when making fishbone diagrams, as the former approach enforce active participation of all involved. Also, the facilitator and form of discussion will still be a bottleneck in terms of productivity. This leads us to conclude that the proposed method of causal maps is less dependent on a professional facilitator, and as such, is more suited for companies who are new to retrospective methods, or where experienced facilitators are not readily available.

The qualitative results presented in Section 6 show that the quality of the analysis when using causal maps is higher than when using fishbone diagrams, in the following respects: the analysis of causes had greater depth; the issues identified were more specific and practical; and the analysis of the cause into subcauses was more varied. We believe that some of these differences are due to the limitations of the structure of the fishbone diagram. It is impractical to analyse fishbone diagrams to a depth of more than three levels. Further, variations in the depth of analysis (from 1-3 levels) are possible but not very practical. Most groups in our experiment conducted their analysis at a depth of two levels for all issues identified. In fishbone diagrams, less relevant issues will be analysed into their component parts, simply to "complete the fishbone structure". In contrast to this, when using causal maps, the structure is constructed after the issues have been identified. Issues that are not very relevant will not be analysed any further, whereas issues that are very relevant will be subject to a more thorough analysis to a depth of several levels. Such analysis will often result in the identification of specific issues that can be addressed with a view to improving performance in future projects. In addition, the construction of causal maps will often yield hubs, which constitute central issues that have several inputs and outputs.

One could argue that there are benefits to using methods that imposing more restrictions on the user, like the fishbone diagram. Afterall the method has been in successful use for a long time. In the process of creating a more restrictive diagram, the user is forced to ask questions, interact and refine their thinking. However, as has been pointed out in previous research [8] and as we have seen in this experiment, this is dependent on an experienced facilitator to properly steer the discussion. When no such facilitators are available, a more freeform technique seems to yield better results.

Another argument often raised against the causal maps, is the concern for "spaghetti diagrams", with no clear structure and the option to connect every item on the map, the diagram might become unreadable and not provide a good starting point for improvement. Fishbone diagrams on the other hand, has a clear structure that makes main causes readily identifiable. In our experiment, however, we did not observe these effects. The causal maps provided good overviews and often had the so called "hubs" which indicated strong causes. The fishbone diagrams on the other hand often did not provide any clear cause, since every bone was filled out "to complete the structure". Once again an indication that an experienced facilitator was needed in this variation.

Each group in the experiment consisted of from four to six persons. We believe that the difference in effectiveness between the two approaches would be even more significant for larger groups. The main reason for this is the form of brainstorming used. A large group using interactive discussion will suffer more from the effect of "production blocking" (impossibility for subjects to speak simultaneously), "evaluation apprehension" (fear of nagative evaluation from other group members), and "free riding" (reduced effort exerted when individual contribution is not identifiable) [13] than a group using a nominal technique. With many subjects present, it is easier to fall silent and leave the discussion to the others. There is a greater risk of the analysis losing focus without coordination of a professional facilitator. The waiting time could also result in a drop of motivation that could hurt the end result.

7.2 Threats to validity

We now discuss possible threats to the validity of our experiment.

7.2.1 Validity of Statistical Conclusions

The hypothesis was tested using both the non-parametric exact Wilcoxon rank sum test and the parametric two-sample t-Test. The tests yielded consistent and significant results. In light of the simplicity of the experiment design and the straightforward statistical analyses, we do not believe that there are major threats to the validity of our statistical conclusions.

7.2.2 Internal Validity

The primary means to address threats to internal validity in this experiment was randomization. In addition, we observed all the PMA sessions to make sure that they conformed fully to the prescribed processes. However, due to practical considerations, once the subjects had been assigned to one of the two treatments, they received different training (on either causal maps or fishbone diagrams, by two different instructors). As explained in Section 4.5, we took several precautions to ensure that the training was as similar as practically possible in quality and quantity, but we cannot completely rule out the possibility that a bias was introduced as a result of this differential training, e.g., that one of the groups became more motivated or better trained in their respective technique than the other group.

7.2.3 Construct Validity

The dependent variable of the experiment was "AnalysisEffectiveness". According to Faure [13], originality of the ideas generated is the most commonly used measure when measuring creative techniques like brainstorming. Note also that the qualitative analyses triangulated the quantitative analysis by offering complementary insights on other aspects of "quality": the qualitative analysis explained and justified the quantitative result.

7.2.4 External Validity

The most prominent threat to external validity is that the experiment was carried out by students for a student project, which is not necessarily representative of industrial settings. However, the students are part of a five-year Masters programme and at the end of their fourth year, when they take the course, many of them have already gained industrial experience as software developers. The project itself was also designed to be as close to a real project as possible, engaging teams of 4-6 developers for a period of four months.

There is also the factor of non-professional facilitators to consider. In previous research on PMA, it has been claimed that the facilitator plays a crucial role [8]. In the experiment, the students had to select a facilitator among themselves. Whether the results can be generalized to a setting with an experienced facilitator, for one or both variants of the method, is a matter for future experiments. We do, however, believe that our results can be generalized to settings in which experienced facilitators are not available.

8 Conclusion

The results of the experiment described in this paper show that when causal maps, rather than fishbone diagrams, are used to analyse successes and/or problems in a PMA, in a setting of small software engineering teams, with no experienced facilitator available, there is a significant increase in both effectiveness and quality. Thus, concerning our first research question: "*Is the revised PMA method more effective than the old PMA method?*", we base our answer on our quantitative analysis which states that there is a statistical significant difference between the two methods and that the effect of using the revised method compared to the old method is *large*. We must also consider the setting of the experiment in our answer, so the final answer is then: Yes, for a setting of small software teams where there is no experienced facilitator available, the revised method is more effective than the original.

To answer research question two: "*How do the two methods differ in their result?*", we used our qualitative observations as well as outlined theory. We conclude that the main explanation for the difference in the two methods is twofold. First, using a nominal brainstorming technique for causal maps will engage the whole evaluation group simultaneously and thus be more productive. This is in line with previous research on brainstorming. Second, the layout of fishbone diagrams limits the ways in which issues can be related and the PMA process can be carried out, and is as such much more dependent on an experienced facilitator to properly steer the discussion. Using fishbone diagrams forces the participants to analyse issues in a strict hierarchical manner and the diagram layout does not encourage deeper analysis into several levels or analysis of the relations between issues. Analysis using causal maps is not restricted in these ways.

The main difference in the use of the two methods was that the use of causal maps produced a more selective and deeper analysis of issues into their component parts that, in many cases, results in the identification of specific and practical issues that can be addressed in order to improve performance in future projects.

The results of our experiment may be extended by performing further experiments, in which the variables and environment are changed. For example, it should be determined how the group size and the usage of a professional facilitator will affect the effectiveness of the variants of the method. To reduce threats to external validity, we should also perform similar experiments in an industrial setting.

References

1. K. Beck. Principles behind the Agile Manifesto. 2001 http://agilemanifesto.org/principles.html. Retrieved May 17th 2007.
2. A. Birk, T. Dingsøyr, and T. Stålhane. Postmortem: Never Leave a Project without it. IEEE Software. 19(3) (2002) 43-45
3. P. Clements, R. Kazman, and M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley Longman Publishing Co., 2002
4. A. Cockburn, Agile Software Development, Addison-Wesley, 2002
5. J. Cohen, Statistical power for the behavioral sciences 2nd ed, Hillsdale, NJ: Erlbaum, 1988
6. C. Collison and G. Parcell, Learning to Fly: Practical Lessons from one of the World's Leading Knowledge Companies, Capstone Publication, 2001
7. K.C. Desouza, T. Dingsøyr, and Y. Awazu. Experiences with Conducting Project Postmortems: Reports versus Stories. Software Process: Improvement and Practice. 10(2) (2005) 203-215
8. T. Dingsøyr. Postmortem reviews: purpose and approaches in software engineering. Information and Software Technology. 47(5) (2005) 293-303
9. T. Dingsøyr and G.K. Hanssen, Extending Agile Methods: Postmortem Reviews as Extended Feedback, Proceedings of the 4th International Workshop on Learning Software Organisations (LSO'02), 2002,
10. T. Dingsøyr, N.B. Moe, and Ø. Nytrø. Augmenting Experience Reports with Lightweight Postmortem Reviews. Lecture Notes in Computer Science. 2188 (2001) 167-181
11. T. Dybå, Enabling Software Process Improvement: An Investigation on the Importance of Organisational Issues, Department of Computer and Information Science, Norwegian University of Science and Technology, 2001
12. T. Dybå, T. Dingsøyr, and N.B. Moe, Process Improvement in Practice - a Handbook for IT Companies, Kluwer, Boston, 2004
13. C. Faure. Beyond Brainstorming: Effects of Different Group Procedures on Selection of Ideas and Satisfaction with the Process. Journal of Creative Behaviour. 38(1) (2004) 13-34
14. R.L. Glass. Project Retrospectives, and Why They Never Happen. IEEE Software. 19(5) (2002) 112-111
15. G.P. Hodgkinson, A.J. Maule, and N.J. Bown. Causal Cognitive Mapping in the Organizational Strategy Field: A Comparison of Alternative Elicitation Procedures. Organizational Research Methods. 7(1) (2004) 3-26
16. A. Keegan and J.R. Turner. Quantity versus Quality in project-based learning practices. Management Learning. 32 (2001) 77-98
17. N.L. Kerth, Project Retrospectives: a handbook for team reviews, Dorset House Publishing, New York, 2001
18. D. Larsen, The Manager's Role in Starting and Ending Projects: Charters and Retrospectives, Proceedings of the 21st Pacific Northwest Software Quality Conference, 2003,

19. M.B. Miles and A.M. Huberman, Qualitative Data Analysis: An expanded sourcebook, second ed., SAGE publications, 1994
20. M. Myllyaho, O. Salo, J. Kääriäinen, and J. Koskela, A Review of Small and Large Post-Mortem Analysis Methods, Proceedings of the ICSSEA, Paris, 2004,
21. L. Rising and E. Derby. Singing the Songs of Project Experience: Patterns and Retrospectives. The Journal of Information Technology Management. 16(9) (2003) 27-33
22. R. Scupin. The KJ Method: a technique for analyzing data derived from Japanese ethnology. Human Organization. 56 (1997) 233-237
23. T. Stålhane, T. Dingsøyr, G.K. Hanssen, and N.B. Moe. Post Mortem - An Assessment of Two Approaches. Lecture Notes in Computer Science. 2765 (2003) 129-141
24. A.I. Wang and T. Stålhane, Using Post Mortem Analysis to Evaluate Software Architecture Student Projects, Proceedings of the 18th Conference on Software Engineering Education and Training, 2005, pp. 43-50.
25. N. Whitten, Managing Software Development Projects: Formula for Success, Wiley, New York, 1995

P7: Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts and Research Methods Used

Submitted Journal of Information and Software Technology, August 2007.

Finn Olav Bjørnson[1] and Torgeir Dingsøyr[1,2]

[1]Norwegian University of Science and Technology Department of Computer and Information Science, Sem Sælandsvei 7-9, 7491 Trondheim, Norway, Tel.:+ 47 73 59 87 16, fax: + 47 73 59 44 66, e-mail: bjornson@idi.ntnu.no

[2]SINTEF Information and Communication Technology, SP Andersens vei 15b, 7465 Trondheim, Norway, Tel.: +47 73 59 29 79, fax: +47 73 59 29 77, e-mail: torgeir.dingsoyr@sintef.no

Abstract. Software engineering is knowledge-intensive work, and how to manage software engineering knowledge has received much attention. This systematic review identifies empirical studies of knowledge management initiatives in software engineering, and discusses the concepts studied and the research methods used. Seven hundred and sixty-two articles were identified, of which 68 were studies in an industry context. Of these, 29 were empirical studies and 39 reports of lessons learned.
The majority of empirical studies relate to technocratic and behavioural aspects of knowledge management, while there are few studies relating to economic, spatial and cartographic approaches. More than half of the empirical studies were case studies.

Keywords: software engineering, knowledge management, learning software organization, software process improvement, systematic review

1. Introduction

In this article, we report on a systematic review of empirical studies of knowledge management in software engineering. Our goal is to provide an overview of empirical studies within this field, what kinds of concepts have been explored, and what research methods are used.

Software engineering is a knowledge-intensive activity. For software organisations, the main assets are not manufacturing plants, buildings, and machines, but the knowledge of the employees. Software engineering has long recognized the need for managing knowledge and the community could learn much from the knowledge-management community, which bases its theories on well-established disciplines such as cognitive science, ergonomics, and management.

As the field of software engineering matures, there is an increased demand for empirically-validated results and not just the testing of technology, which seems to have dominated the field so far. A recent trend in software engineering is an increased focus on evidence-based software engineering, EBSE [37, 59]. Since the volume of research in the field is expanding constantly, it is becoming more and more difficult to evaluate critically and to synthesise the material in any given area. This has lead to an increased interest in systematic reviews (SR) [58] within the field of software engineering.

The purpose of this paper is (1) to perform a systematic review of empirical studies of knowledge management in software engineering, (2) to present the major concepts that has been investigated and the research methods used, and (3) to point out potential research gaps in the field that require further investigation.

Our target readership is three groups that we think will be interested in an overview of empirical research on knowledge management in software engineering: (1) researchers from software engineering who would also be interested in what concepts have been researched, and how these concepts have been researched; (2) researchers on knowledge management in general, who would be interested in comparing work in the software engineering field to other knowledge-intensive fields; and (3) reflective practitioners in software engineering, who will know what knowledge management initiatives have been made in software companies.

The remainder of this article is structured as follows. Section 2 presents the background and general theories on knowledge management. Section 3 describes the research method that we used to select and review the data material for our research, and presents our chosen framework for analysis. Section 4 presents the results of the systematic review according to our chosen framework. In Section 5, we discuss the implications of our findings. Section 6 concludes.

2. Background

2.1 Knowledge management

Knowledge management is a large interdisciplinary field. There is, as a consequence, an ongoing debate as to what constitutes knowledge management. However, it is beyond the scope of this article to engage in that debate. For our purposes, it is sufficient to cite some definitions that are in common use. Davenport has defined *knowledge management* as ”a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company’s knowledge” [23]. A related term is *organisational learning*. What does it mean to say that an organisation as a whole learns? According to Stata, this differs from individual learning in two respects [99]: first, it occurs through shared insight, knowledge and shared models; second, it is based not only on the memory of the participants in the organisation, but also on “institutional mechanisms” such as policies, strategies, explicit models and defined processes (we can call this the “culture” of the organisation). These mechanisms may change over time, what we can say is a form of learning.

Knowledge management has received much attention in various fields, which is shown through two “handbooks” [28, 39], one encyclopaedia [94], and numerous books [21, 23, 97].

Hanssen et al. [49] refer to two main strategies for knowledge management:

- Codification – to systematise and store information that constitutes the knowledge of the company, and to make this available to the people in the company.
- Personalisation – to support the flow of information in a company by having a centralised store of information about knowledge sources, like a ”yellow pages” of who knows what in a company.

Earl [38] has further classified work in knowledge management into schools (see Table 1). The schools are broadly categorized as “technocratic”, “economic” and “behavioural”. The technocratic schools are 1) the systems school, which focuses on technology for knowledge sharing, using knowledge repositories; 2) the cartographic school, which focuses on knowledge maps and creating knowledge directories; and 3) the engineering school, which focuses on processes and knowledge flows in organizations.

The economic school focuses on how knowledge assets relates to income in organizations.

The behavioural school consists of three subschools: 1) the organizational school, which focuses on networks for sharing knowledge; 2) the spatial school, which focuses on how office space can be designed to promote knowledge sharing; and 3) the strategic school, which focuses on how knowledge can be seen as the essence of a company’s strategy.

Table 14: Earl's schools of knowledge management.

	Technocratic			Economic	Behavioural		
	Systems	Cartographic	Engineering	Commercial	Organizational	Spatial	Strategic
Focus	Technology	Maps	Processes	Income	Networks	Space	Mindset
Aim	Knowledge bases	Knowledge directories	Knowledge flows	Knowledge assets	Knowledge pooling	Knowledge exchange	Knowledge capabilities
Unit	Domain	Enterprise	Activity	Know-how	Communities	Place	Business

There are a number of overview articles of the knowledge management field in the literature. Alavi et al. [3] give an overview of the knowledge management literature in different fields. They identify research issues in knowledge management related to knowledge creation, storage and retrieval of knowledge, knowledge transfer, and knowledge application.

Liao gives an overview of technology and applications for knowledge management in a review of the literature from 1995 to 2002 [66].

Argote et al. [7] conclude a special issue of Management Science with an article that provides a framework for organizing the literature on knowledge management, identifies emerging themes, and suggests directions for further research.

In Section 2.2, we give an overview of theories often referred to in the knowledge management literature. In Section 2.3, we give an overview of existing work on knowledge management in software engineering.

2.2 Theories of learning

In cognitive and organization science, we find many models on how knowledge is transferred or learned at an individual and organizational level. We present four theories that are referred to widely: Kolb's model of experiential learning, the double-loop learning theory of Argyris and Schön, Wenger's theory of communities of practice, and Nonaka and Takeuchi´s theory of knowledge creation.

Kolb describes learning from experience ("experiential learning", see [62]) as four different learning modes that we can place in two dimensions. One dimension is how people take hold of experience, with two modes, either relying on symbolic representation – which he calls comprehension, or through "tangible, felt qualities of immediate experience", which he calls apprehension. The other dimension is how people transform experience, with two modes, either through internal reflection, which he refers to as intention, or through "active external manipulation of the external world", which he calls extension.

Kolb argues that people need to take advantage of all four modes of learning to be effective, they "must be able to involve themselves fully, openly, and without bias in new experiences; reflect on and observe these experiences from many perspectives; create concepts that integrate their observations into logically sound theories; and use these theories to make decisions and solve problems" [63].

Argyris and Schön distinguish between what they call single and double-loop learning [9] in organisations. In single-loop learning, one receives feedback in the form of observed effects and then acts on the basis solely of these observations to change and improve the process or causal chain of events that generated them. In double-loop learning, one not only observes the effects of a process or causal chain of events, but also understands the factors that influence the effects [8].

One traditional view of learning is that it best takes place in a setting where you isolate and abstract knowledge and then "teach" it to "students" in rooms free of context. Wenger describes this as a view of learning as an individual process where, for example, collaboration is considered a kind of cheating [106]. In his book about communities of practice, he describes a completely different view: learning as a *social phenomenon*. A community of practice develops its own "practices, routines, rituals, artefacts, symbols, conventions, stories and histories". This is often different from what you find in work instructions, manuals and the like. Wenger defines learning in communities of practice as follows:

For individuals: learning takes place in the course of engaging in, and contributing to, a community.
For communities: learning is to refine the practice.
For organisations: learning is to sustain interconnected communities of practice.

Nonaka and Takeuchi [79] claim that knowledge is constantly converted from tacit to explicit and back again as it passes through an organisation. By tacit knowledge [83] we mean knowledge that a human is not able to express explicitly, but is guiding the behaviour of the human. Explicit knowledge is knowledge that we can represent in textual or symbolic form. They say that knowledge can be converted from tacit to tacit, from tacit to explicit, or from explicit to either tacit or explicit knowledge. These modes of conversion are described as follows:

Socialization means to transfer tacit knowledge to another person through observation, imitation and practice, what has been referred to as "on the job" training. *Externalisation* means to go from tacit knowledge to explicit. Explicit knowledge can "take the shapes of metaphors, analogies, concepts, hypotheses or models". *Internalisation* means to take externalised knowledge and make it into individual tacit knowledge in the form of mental models or technical know-how.
Combination means to go from explicit to explicit knowledge, by taking knowledge from different sources such as documents, meetings, telephone conferences, or bulletin boards and aggregating and systematizing it.

According to Nonaka and Takeuchi, knowledge passes through different modes of conversion, which makes the knowledge more refined and spreads it across different layers in an organisation.

2.3 Knowledge management in software engineering

In software engineering, there has been much discussion about how to manage knowledge, or foster “learning software organisations”. In this context, Feldmann and Althoff have defined a “learning software organisation” as an organisation that has to “create a culture that promotes continuous learning and fosters the exchange of experience” [44]. Dybå places more emphasis on action in his definition: “A software organisation that promotes improved actions through better knowledge and understanding” [35].

In software engineering, reusing life cycle experience, processes and products for software development is often referred to as having an “Experience Factory” [13]. In this framework, experience is collected from software development projects, and are packaged and stored in an *experience base*. By packing, we mean generalising, tailoring, and formalising experience so that it is easy to reuse.

In 1999, the first workshop on “learning software organizations” was organized in conjunction with the SEKE conference. This workshop has been one of the main arenas for empirical studies as well as technological development related to knowledge management in software engineering.

The May 2002 issue of IEEE Software [69] was devoted to knowledge management in software engineering, giving several examples of knowledge management applications in software companies. In 2003, the book “Managing Software Engineering Knowledge” [40] was published, focusing on a range of topics, from identifying why knowledge management is important in software engineering [70], to supporting structures for knowledge management applications in software engineering, to offering practical guidelines for managing knowledge.

However, Edwards notes in an overview chapter in the book on Managing Software Engineering Knowledge [41] that knowledge management in software engineering is somewhat distanced from mainstream knowledge management.

Several PhD thesis have also been published on aspects of knowledge management that are related to software engineering [15, 31, 103].

In addition, a number of overviews of work on knowledge management in software engineering have previously been published. Rus et al. [89] present an overview of knowledge management in software engineering. The review focuses on motivations for knowledge management, approaches to knowledge management, and factors that are important when implementing knowledge management strategies in software companies. Lindvall et al. [72] describe types of software tools that are relevant for knowledge management, including tools for managing documents and content, tools for managing competence, and tools for collaboration. Dingsøyr and Conradi [32] surveyed the literature for studies of knowledge management initiatives in software engineering. They found eight reports on lessons learned, which are formulated with respect to what actions companies took, what the effects of the actions were, what benefits are reported, and what kinds of strategy for managing knowledge were used.

Despite of the previously published overviews of the field, there is still a lack of broad overviews of knowledge management in software engineering. Our motivation for this study was thus, to give a more thorough and broader overview in the form of a systematic review. This study also covers recent work, and assesses the quality of the research in the field.

3. Method

The research method used is a systematic review [58], with demands placed on research questions, identification of research, selection process, appraisal, synthesis, and inferences. We now address each of these in turn.

3.1 Planning the review

We started by developing a protocol for the systematic review, specifying in advance the process and methods that we would apply. The protocol specified the research questions, the search strategy, criteria for inclusion and exclusion, and method of synthesis.

The aim of the study was to provide an overview of the empirically studied methods for knowledge management in software engineering. To achieve this, we decided to address the following research questions:

(1) What concepts have been investigated empirically within the field of knowledge management in software engineering.?
(2) What are the research methods used in studying knowledge management in software engineering?

3.2 Identification of research

A comprehensive, unbiased search is a fundamental factor that distinguishes a systematic review from a traditional review of the literature. Our systematic search started with the identification of keywords and search terms.

Table 15: Keywords for our search

Software engineering keywords	Knowledge management keywords
• software engineering • software process • learning software organization	• knowledge management • tacit knowledge • explicit knowledge • knowledge creation • knowledge acquisition • knowledge sharing • knowledge retention

	• knowledge valuation • knowledge use • knowledge application • knowledge discovery • knowledge integration • knowledge Theory • organization knowledge • knowledge engineering • experience transfer • technology transfer

All possible permutations of the software engineering and knowledge management concepts were tried in the search conducted.

The following electronic bases were searched, using the strategy outlined. The electronic bases were those we considered most relevant [36]: ISI Web of Science, Compendex, IEEE Xplore and the ACM Digital Library.

In addition, we identified two arenas that, to our knowledge, are the only ones that pertain specifically to knowledge management in software engineering: the workshop series on Learning Software Organisations (LSO) from 1999 until 2006, and the book Managing Software Engineering Knowledge [10]. We searched all proceedings from the workshop series and included all chapters from the book.

We performed the search in August 2006, which means that publications up to and including the first quarter of 2006 are included, but some studies in the second quarter might not have been indexed in the databases.

The identification process yielded 2102 articles. This formed the basis for the next step in our selection process.

3.3 Selection of primary studies

The first step after the articles had been identified was to eliminate duplicate titles, and titles clearly not related to the review. One researcher (FOB) read through the 2102 titles and removed duplicates and those clearly not related to the field of software engineering. This yielded a result of 762 articles.

After this we obtained the abstract of these articles and both authors read through all abstracts, with the following exclusion criterion.

- Exclude if the focus of the paper is clearly not on software engineering
- Exclude if the focus of the paper is clearly not on knowledge management
- Exclude if the method, tool or theory described is not tested in industry

To narrow the search further we also decided to focus on technical and process knowledge (thus, “software engineering knowledge”). Hence, we also used the criterion

- Exclude if the focus of the paper is on domain knowledge

After each researcher had gone through the papers we compared results. Where we disagreed as to whether to keep or remove a paper, we discussed the matter until we reached agreement.

This process reduced the number of articles to 133, and agreement between researchers was ‘good’ (Kappa value of 0,655).

The full text for all 133 papers was obtained and both researchers read through all the papers with the same criteria for exclusion in mind. The final number of papers selected for the review was 68. The agreement between researchers at this stage was “moderate” (Kappa value: 0,523).

3.4 Quality assessment and classification

We chose to classify the 68 papers identified along two axes. (1) We wanted to examine what kinds of concept had been tested. To aid us with this we chose the framework for classifying strategies for managing knowledge presented by Earl in [38]. Each researcher classified the 68 papers individually according to the framework, before comparing the results. Disagreements were discussed until a consensus was reached on the classification. (2) We also wanted to examine the scientific rigour of the studies. Here we settled on a simpler classification. All studies included so far had results taken from industry. We further assessed the quality of the selected papers by categorising these into empirical studies and lessons learned reports. The criterion for being accepted as an empirical study and not a report of lessons learned was that the article had a section describing the research method and context. Again, each study was classified individually by the two researchers before comparing the results and discussing problem cases in order to reach agreement. After the quality assessment, we had 29 empirical studies and 39 reports of lessons learned.

3.5 Synthesis

For the synthesis, we used the papers classified as empirical studies in our framework. We extracted concepts covered and the research method for each article. One researcher (FOB) focused on the studies in the technocratic schools, while the other researcher (TD) focused on the behavioural schools.

4. Results

We found a total of 68 papers that we considered to lie within our scope for this review, 29 of which we considered to be of sufficient quality to be categorized as empirical studies and 39 as reports of lessons learned. The result of our categorization is presented in Table 3. For a complete listing of papers in each category, see the appendix. Within Earl’s framework, we found a heavy concentration on the technocratic schools and a fair mention of the behavioural school. We did not find any papers relating to the commercial school with our search criterion. Within the technocratic schools, systems and engineering stand out as areas that have received much attention. Within the behavioural schools, organizational and strategic have received the most attention.

Four of the empirical studies did not fit into Earl's framework. These were classified as studies on the impact of knowledge management initiatives and on knowledge management per se. Thus, we ended up with 25 studies classified as empirical within the framework. Of the 39 reports of lessons learned, two belonged to two categories, which is why we ended up with a sum of 41 for the reports of lessons learned in the table.

Table 16: Categorized articles

	Systems	Cartographic	Engineering	Commercial	Organizational	Spatial	Strategic	SUM
Empirical studies	6	1	12	0	3	0	3	25
% distribution, empiricla studies	*24*	*4*	*48*	*0*	*12*	*0*	*12*	100
Lessons learned reports	20	0	9	0	2	1	9	41
% distribution, lessons learned reports	*49*	*0*	*22*	*0*	*5*	*2*	*22*	100

Looking at the papers by year of publication, presented in Figure 1, we notice an increasing interest in the area from 1999 onwards. We also notice a shift from more papers on lessons learned to empirical papers from 2003 onwards. The apparent decrease in attention in 2006 is due to our covering only the first third of this year, since our search was conducted in August.

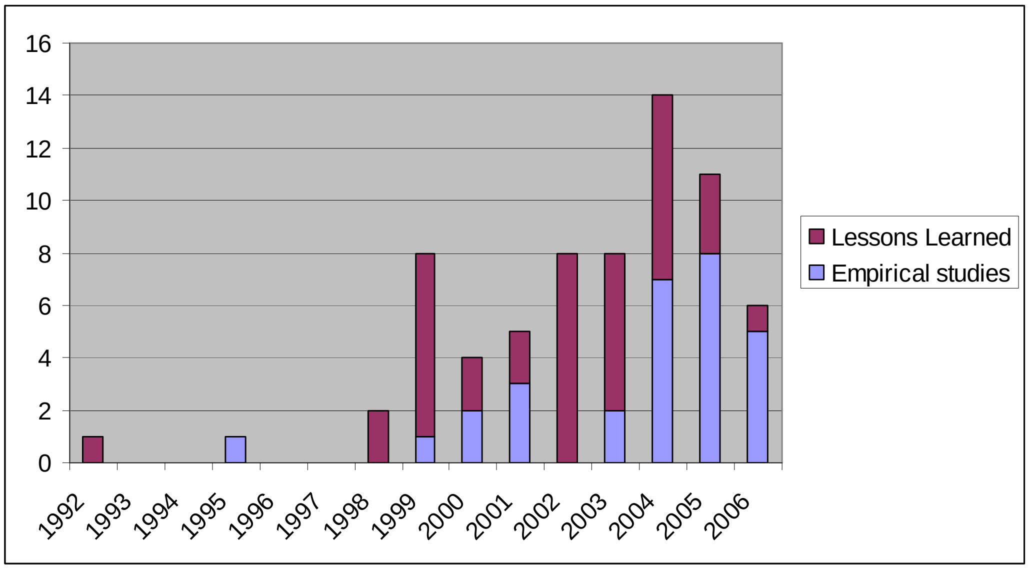

Figure 12: Publications by year

To obtain an overview of the research methods used within this field, we used the classification presented in Glass et al. [46]. This was carried out on the 25 papers classified as empirical studies. The result is presented in Table 4. See the appendix for a complete listing of which paper was classified in which category.

Table 17: Overview of research methods

	Action Research	Case study	Etnography	Experiment	Field study	Sum
Systems	1	3	1		1	6
Cartographic			1			1
Engineering	1	8		1	2	12
Organizational		3				3
Strategic	1				2	3
Sum	3	14	2	1	5	25
%	12	56	8	4	20	100

In the following subsections, we briefly present the context and concepts within our major categories.

4.1 Technocratic schools

The technocratic schools are based on information or management technologies, which largely support and, to different degrees, condition employees in their everyday tasks. We identified a total of 19 empirical studies and 29 papers on lessons learned in this category. The main focus is on the engineering and systems schools.

4.1.1 Systems

As defined by Earl, the systems school is built on the underlying principle that knowledge should be codified in knowledge bases. This is what Hansen et al. refer to as the "codification strategy", and what Nonaka and Takeuchi refer to as externalization.

This school is the longest established school of knowledge management, and it is in this category we found the oldest papers in our search. Most of the papers that were excluded would have been placed in this category, if they had contained empirical results from industry. They could mainly be classified as conceptual analysis and concept implementation, according to Glass's definition. In total, we classified six papers as empirical in this school, and 20 as lessons learned. The empirical papers in this category can broadly be defined as either dealing with the development of knowledge bases or the use of such bases. In what follows, we briefly present the major concepts studied in the empirical papers.

In [20], Chewar and McCrickard present their conclusions from three case studies investigating the use of their knowledge repository. On the basis of their case studies, they present general guidelines and tradeoffs for developing a knowledge repository. In [17], Bjørnson and Stålhane follow a small consulting company that wanted to introduce an experience repository. On the basis of interviews with the employees, they draw conclusions about attitudes towards the new experience repository, and the content and functionality preferred by the employees. Barros et al. [11] investigate how risk archetypes and scenario models can be used to codify reusable knowledge about project management. They test their approach by an observational analysis in industry. They also describe a feasibility study within an academic environment.

Concerning the actual usage of experience repositories or knowledge bases, Dingsøyr and Røyrvik [30] investigate the practices in a medium-sized software consulting company where knowledge repositories are used in concrete work situations. They found several distinct ways of using the tool and highlight the importance of informal organization and the social integration of the tool in daily work practices. A more formal approach to knowledge management tools is found in [98], where Skuce describes experiences from applying a knowledge management tool in the design of a large commercial software system. Concerning long-term effects of experience repositories, Kurniawati and Jeffrey [64] followed the usage of a combined electronic process guide and experience repository in a small-to-medium-sized software development company for 21 weeks, starting a year after the tool was introduced. They conclude that tangible benefits can be realized quickly and that the tool remains useful with more benefits accruing over time.

4.1.2 Cartographic

The principal idea of the cartographic school is to make sure that knowledgeable people in an organization are accessible to each other for advice, consultation, or knowledge exchange. This is often achieved through knowledge directories, or so-called ”yellow pages”, that can be searched for information as required.

We found only one empirical paper within this school and no papers on lessons learned. In [29], Dingsøyr et al. examine a skills management tool at a medium-sized consulting company. They identify four major usages of the tool and point out implications of their findings for future or other existing tools in this category.

4.1.3 Engineering

The engineering school of knowledge management is a derivative or outgrowth of business process reengineering. Consequently it focuses on processes. According to our classification, the largest amount of empirical papers came from this school. Two major categories can be identified. The first contains work done by researchers who investigate the entire software process with respect to knowledge management. The second contains work done by researchers who focus more on specific activities and how the process can be improved within this activity.

Baskerville and Pries-Heje [14] used knowledge management as the underlying theory to develop a set of key process areas to supplement the Capability Maturity Model (CMM) [82] in a Small and Medium sized Enterprise (SME) software development company. Realising that the CMM did not fit well with an SME company, they helped their case companies to develop new key process areas that focused on managing their knowledge capability. Arent et al. [6] address the challenge of creating organizational knowledge during software process improvement. They argue for the importance of creating organizational knowledge in Software Process Improvement (SPI) efforts and claim that its creation is a major factor for success. On the basis of an examination of several cases, they claim that both explicit and tacit knowledge are required, no matter what approach is pursued. Segal [96] investigates organizational learning in software process improvement. Using a case to initiate and implement a manual of best practice as a basis, she observed that the ideal and actual scenarios of use differed and identified possible reasons for the difference. In [45] Folkestad et al. studied the effect of using the rational unified process as a tool for organizational change. In this case, it was used to introduce development staff to a new technology and methodology. Folkestad et al. concluded that the iterative approach of the unified process had obvious effects on organisational and individual learning. The unified process also resulted in new patterns of communication and a new division of labour being instituted, which had a significant effect on the company. Wangenheim et al. [104] report on their experiences of defining and implementing software processes. They confirm what others have experienced, that it is possible to define and implement software processes in the context of small companies in a beneficial and cost-effective way.

In the papers that focused on specific activities within the process, we identified four major areas: formal routines, mapping of knowledge flows, project reviews, and social interaction. Many of these processes are aimed at stimulating several ways of learning, as, for example, Kolb suggests.

In [22] Conradi and Dybå report on a survey that investigated the utility of formal routines for transferring knowledge and experience. Their main observation was that developers were rather sceptical about using written routines, while quality and technical managers took this for granted. Given this conflict of attitudes, they describe three implications for research on this topic.

Hansen and Kautz [48] argue that if software companies are to survive, it is critical that they improve continuously the services that they provide. Such improvement depends, to a great extent, on the organization’s capability to share knowledge and thus on the way knowledge flows in an organization. To investigate knowledge flow, they introduced a tool to map the flows of organisational knowledge in a software development company. Using their new method, they identify potential threats to knowledge flows in an organisation. Also using flow diagrams, Al-Shehab et al. [2] describe how learning from analyses of past projects and from the issues that contributed to their failure is becoming a major stage in the risk management process. They introduce causal mapping as a method to visualise cause and effect in risk networks. They claim that their method is useful for organisational learning, because it helps people to visualise differences in perceptions.

In [27], Desouza et al. describe two ways of conducting project postmortems. They stress that learning through postmortems must occur at three levels: individual, team, and organization. The paper describes guidelines for when to select different kinds of postmortem, depending on the context and the knowledge that is to be shared. The authors also argue that postmortems must be woven into the fabric of current project management practices. Salo [90] also studies postmortem techniques and concludes that existing techniques lack a systematic approach to validating iteratively the implementation and effectiveness of action taken to improve software processes. Salo studies the implementation of a method to remedy this and observes that the organisational level can only benefit from the learning of project teams if the knowledge and reasoning behind the improvements to processes are converted into an explicit format such that it can be utilized for learning at the organisational level.

In [76], Melnik and Maurer discuss the role of conversation and social interaction effective knowledge sharing in an agile process. Their main finding suggests that the focus on pure codification is the principal reason that Tailoristic teams fail to share knowledge effectively. Moving the focus from codification to socialisation, Bjørnson and Dingsøyr [16] investigated knowledge sharing through a mentor programme in a small software consultancy company. They describe how mentor programmes could be changed to improve the learning in the organization. They also identify several unofficial learning schemes that could be improved.

4.2 Behavioural schools

The behavioural aspects of knowledge management are covered in three schools in Earl's framework: the organizational, spatial, and strategic schools. In our review, we found three empirical studies and two reports of lessons learned in the organizational school, no empirical study and one report of lessons learned in the spatial school, and three empirical studies and nine reports of lessons learned in the strategic school. We present the main concepts from the organizational and strategic schools.

4.2.1 Organizational

The organizational school focuses on describing the use of organizational structures (networks) to share or pool knowledge. These structures are often referred to as "knowledge communities". Work on knowledge communities is related to work on communities of practice as described in Section 2.2.

The role of networking as an approach to knowledge management has been investigated in three settings where software is developed. Grabher and Ibert [47] discuss what types of network exist in companies, where one case is a software company based in Germany. Mathiassen and Vogelsang [75] discuss how to implement software methods in practice and use two concepts from knowledge management: networks and networking. The network perspective emphasizes the use of technology for sharing knowledge, while networking focuses on trust and collaboration among practitioners involved in software development. Nörbjerg et al. [80] discuss the advantages and limitations of knowledge networks. They base their discussion on an analysis of two networks related to software process improvement in a medium-sized software company in Europe.

4.2.2 Strategic

In the strategic school, knowledge management is seen as a dimension of competitive strategy. Skandia’s views are a prime example [100]. Developing conceptual models of the purpose and nature of intellectual capital has been a central issue.

One important issue in the literature on knowledge management has been to identify the factors that lead to the successful management of knowledge. Feher and Gabor [43] developed a model of the factors that support knowledge management. The model was developed on the basis of data on 72 software development organizations that are contained in the European database for the improvement of software processes.

Another issue of strategic importance is the processes that are in place to facilitate learning. Arent and Nørjeberg [5] analysed three industrial projects for the improvement of software processes, in order to identify the learning processes used. They found that both tacit and explicit knowledge were important for improving practice, and that improvement requires ongoing interaction between different learning processes.

Trittmann [102] distinguish between two types of strategy for managing knowledge: “mechanistic” and ”organic”. Organic knowledge management pertains to activities that seek to foster innovation, while mechanistic knowledge management aims at using existing knowledge. A survey of 28 software companies in Germany supported the existence of two such strategies. This work parallels the works of Hansen et al. on codification and personalization as important strategies for managing knowledge in the field of management science.

4.3 Knowledge management in general

Some studies could not be classified using Earl’s framework. These studies can be placed in a broad category that encompasses works that seek to identify the impact of knowledge management initiatives (two empirical studies), and works that investigate knowledge management per se (two empirical studies).

4.3.1 The impact of knowledge management initiatives

Ajila and Sun [1] investigated two approaches to delivering knowledge to software development projects: ”push” and ”pull”. “Push” means using tools to identify and provide knowledge to potential users. “Pull” means that users themselves have to use repositories and other tools to identify relevant knowledge. On the basis of a survey of 41 software companies in North America, the authors claim that pulling leads to more effective software development.

Ravichandran and Rai [86] studied two models for how the embedding and creation of knowledge influence software process capability. Embedding refers to the process of employing knowledge in standard practices, for example through making work routines, methods and procedures. They found support for a model where knowledge creation has an effect on process capability when the knowledge is embedded after it is created. This means that knowledge has to be internalized before it can be used to improve processes. The study was done as a survey of 103 Fortune 1000 companies and federal and state government agencies in the US.

4.3.2 Knowledge management per se

Ward and Aurum [105] describe current practices for managing knowledge in two Australian software companies and explain how leadership, technology, culture, and measurements enable knowledge to be managed effectively and efficiently. They found leadership to be the most significant positive factor for the management of knowledge, but that the tools, techniques, and methodologies that the companies were using were not adequate for managing knowledge effectively.

Desouza et al. [26] examined what factors contribute to the use of knowledge artefacts in a survey of 175 employees in a software engineering organization. They specifically looked at factors that govern the use of explicit knowledge. They found that the following factors relate to the use of explicit knowledge: perceived complexity, perceived relative advantage, and perceived risk.

5. Discussion

We now discuss our findings. We begin with a discussion concerning our two research questions, and end with a discussion of the validity of our study.

5.1 Concepts

In answering our first research question regarding concepts investigated empirically within the field, we decided to use Earl's framework for schools of knowledge management. The final selection of papers was divided between the technocratic and behavioural schools, with an emphasis on the technocratic side. This was not surprising, given the general focus of software engineering on the construction of tools and processes. We did not find any examples of what Earl considers economic schools. The reason for this is probably that not many software companies track their intellectual capital.

Looking closer at the technocratic schools, we saw a heavy focus on the systems and engineering schools, with barely any mention of the cartographic school. The heavy focus on the systems school can be explained by the software engineering field's focus on implementing new tools. The ratio of empirical versus lessons-learned papers also confirmed what has been pointed out previously; that there is a heavy focus on building new tools, but far too little on testing and reporting the actual usage of these tools. As mentioned previously, many of the excluded papers would have been placed in this category, had they had any empirical content. The main concepts we identified in this school were the development and use of knowledge repositories. There was, however, little to no overlap between the identified papers, which underlines once again the need for more empirical research.

The engineering school is the school that received the most empirical attention, according to our review. Again, we identified two main areas within this school: those focusing on the entire software process and those focusing on particular activities within the process. Within the papers focusing on specific activities, we identified four main areas: formal routines, mapping of knowledge flows, project reviews, and social interaction. As with the systems school, there is little or no overlap between the empirical studies. A possible explanation for the heavy empirical focus within this school is the close fit with work on the improvement of software development processes.

That there are so few papers in the cartographic school is interesting. One possible explanation is that the ”yellow pages” systems are considered ”simple” and undeserving of attention. However, as the lone study in this category shows, such tools have uses other than the obvious. This school could benefit from more studies of actual usage.

In the behavioural school, we found a limited number of papers focusing on organizational and strategic aspects, and no papers focusing on spatial aspects.

The three studies in the organizational school discuss the use of people networks in software organizations. Two of the studies investigated the improvement of software development processes. In Earl’s taxonomy, both intra- and interorganizational communities are mentioned as examples. In the software engineering literature, we only find studies made in single organizations.

As for the spatial school, no empirical studies on software engineering were found in this category. This is clearly an area where more research should be conducted. The role of open-plan offices has been studied in other fields, and this is something that also should have an impact on how knowledge is shared in software teams. Many of the agile development methods recommend open-plan offices.

The empirical studies in the strategic school focus on factors pertaining to successful knowledge management, learning processes, and types of strategy for managing knowledge. It was, perhaps, to be expected that there would not be many articles discussing the strategic importance of knowledge in software engineering supported by empirical findings, because its importance is assumed in most published works on knowledge management in software engineering.

5.2 Research methods

Of the 68 studies identified, 39 were reports of lessons learned and 29 were empirical studies. Case studies constituted the largest number of empirical studies (see Table 4), followed by field studies and action research. It is positive that the emphasis on empirical studies has increased (see Figure 1). The apparent dip in 2006 is due to the time at which the search was conducted. We searched the databases in August and most compilers of databases take some months to index their papers; hence, we can only claim to have covered the first third of 2006 fully.

The research methods in the studies that we selected are dominated by case studies, both single and multiple. This is not surprising, considering our limitation on only including studies that performed tests in industry.

Glass et al. [46] found that empirical studies constitute about 5% of published research in software engineering as a whole. Comparing our final findings to the results from our first rough sorting of papers, our final selection constituted about 3% of the initially selected papers. If we assume that Glass's data are representative for the area that we studied within software engineering, we could extrapolate that about 70% of those papers would be conceptual analysis and concept implementation. Most of the papers discarded were indeed conceptual analysis and concept implementation without empirical testing, our results do however, not show a discard number on the empirical criterion as high as 70%. Many studies were also excluded because they were not relevant to either software engineering or knowledge management. Therefore it seems that empirical studies constitute a larger part of the studies on knowledge management in software engineering than in software engineering in general.

5.3 The state of research on knowledge management in software engineering

We identified far more studies, particularly empirical studies, than have been reported in previous assessments by Rus et al. [89], Lindvall [72] and Dingsøyr and Conradi [32]. We have also shown that although there are not many empirical studies, except for in the systems and engineering schools, there are either empirical studies or reports of lessons learned in all schools except the economic school. Thus, research on knowledge management in software engineering seems to be slowly gaining a broader focus, although research on knowledge management in software engineering is still somewhat distanced from mainstream research on knowledge management.

If we compare the studies found in software engineering to the research directions suggested by Alavi et al. [3], we see that software engineering has primarily addressed the storage and retrieval of knowledge, while topics such as knowledge creation the transfer and application of knowledge still needs more attention.

5.4 Limitations

The main threats to validity in this systematic review are threefold: our selection of the studies to be included, the classification of studies according to Earl's framework of schools in knowledge management, and potential author bias.

As for the selection of studies, only one researcher read through and discarded the first results on the basis of the papers' titles. However, in cases where there was doubt, the papers were included in the next stage. The second and third stages, which were based on abstracts and full papers, were carried out by both researchers and we observed a 'good' degree of consensus. In cases where there was disagreement, the issue was discussed until concensus was reached.

Finally, there is a potential bias in that both authors have written papers that were included in the review. Where only one author had participated in the primary study, the other author decided whether or not to include it if there was disagreement.

6. Conclusion

This systematic review has addressed the following research questions. 1) Which concepts have been investigated empirically within the field of knowledge management in software engineering? 2) What are the research methods used in studying knowledge management in software engineering?

For the first research question, our main findings are:

- The majority of studies of knowledge management in software engineering relate to technocratic and behavioural aspects of knowledge management.
- The studies that report on concepts within the fields of technocratic and behavioural aspects have very little overlap.
- There are few studies relating to economic, spatial and cartographic approaches to knowledge management.

For the second research question, we found that:

- The majority of reports of applications of knowledge management in the software engineering industry are reports of lessons learned, not scientific studies.
- Of the reports categorized as empirical studies, more than half of the reports are case studies.
- Our search returned field studies, action research, ethnographic studies, and one laboratory experiment.

We see a clear need for more empirical studies of knowledge management in software engineering, especially in the areas that have so far received little attention. There should also be more primary studies carried out on the effects of the approaches that are used in the software industry. These studies are needed in order to understand how knowledge is shared in software companies, and also to offer better advice on what works to the software industry.

Acknowledgement
We are grateful to Reidar Conradi at the Department of Computer and Information Science, Norwegian University of Science and Technology, and Tore Dybå at SINTEF ICT for comments on an earlier version of this article. We would also like to thank Chris Wright for proofreading and useful comments.

References

1. S.A. Ajila and Z. Sun, Knowledge management: Impact of knowledge delivery factors on software product development efficiency, Proceedings of the IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, United States, 2004, pp. 320-325.

2. A.J. Al-Shehab, R.T. Hughes, and G. Winstanley, Facilitating Organisational Learning Through Causal Mapping, Proceedings of the 7th International Workshop on Learning Software Organizations, Springer Verlag, Kaiserslautern, Germany, 2005, pp. 145-154.
3. M. Alavi and D.E. Leidner. Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. MIS Quarterly. 25(1) (2001) 107-136
4. N. Angkasaputra, D. Pfahl, E. Ras, and S. Trapp, The Collaborative Learning Methodology CORONET-Train: Implementation and Guidance, Proceedings of the 4th International Workshop on Learning Software Organizations, Springer Verlag, Chicago, IL, USA, 2002, pp. 13-24.
5. J. Arent and J. Norbjerg, Software process improvement as organizational knowledge creation: a multiple case analysis, Proceedings of the Hawaii International Conference on System Sciences, Maui, USA, 2000, pp. 105.
6. J. Arent, J. Nørbjerg, and M.H. Pedersen, Creating Organizational Knowledge in Software Process Improvement, Proceedings of the 2nd Workshop on Learning Software Organizations, Oulu, Finland, 2000, pp. 81-92.
7. L. Argote, B. McEvily, and R. Reagans. Managing Knowledge in Organizations: An Integrative Framework and Review of Emerging Themes. Management Science. 49(4) (2003) 571-582
8. C. Argyris, Overcoming Organizational Defences: Facilitating Organizational Learning, Prentice Hall, 1990
9. C. Argyris and D.A. Schön, Organizational Learning II: Theory, Method and Practise, Addison Wesley, 1996
10. A. Aurum, R. Jeffrey, C. Wohlin, and M. Handzic, Managing Software Engineering Knowledge, Springer-Verlag, 2003
11. M.d.O. Barros, C.M.L. Werner, and G.H. Travassos. Supporting risks in software project management. Journal of Systems and Software. 70(1-2) (2004) 21-35
12. V.R. Basili, G. Caldiera, F. McGarry, R. Pajerski, and G. Page, The Software Engineering Laboratory - An operational software experience factory, Proceedings of the 14th International Conference on Software Engineering, 1992, pp. 370-381.
13. V.R. Basili, G. Caldiera, and H.D. Rombach, The Experience Factory, in: J.J. Marciniak (Eds.), Encyclopedia of Software Engineering, 1, John Wiley, 1994, pp. 469-476.
14. P.-H.J. Baskerville Richard. Knowledge capability and maturity in software management (1999
15. A. Birk, A Knowledge Management Infrastructure for Systematic Improvement in Software Engineering, Dr. Ing thesis, University of Kaiserslautern, Department of Informatics, 2000
16. F.O. Bjornson and T. Dingsoyr, A study of a mentoring program for knowledge transfer in a small software consultancy company, Proceedings of, Springer Verlag, Heidelberg, D-69121, Germany, Oulu, Finland, 2005, pp. 245-256.
17. F.O. Bjørnson and T. Stålhane, Harvesting Knowledge through a Method Framework in an Electronic Process Guide, Proceedings of the 7th International

Workshop on Learning Software Organizations, Springer Verlag, Kaiserslautern, Germany, 2005, pp. 86-90.

18. P. Brössler, Knowledge Management at a Software Engineering Company - An Experience Report, Proceedings of the 1st Workshop on Learning Software Organizations, Kaiserslautern, Germany, 1999, pp. 77-86.

19. B. Chatters, Implementing an experience factory: maintenance and evolution of the software and systems development process, Proceedings of, 1999, pp. 146-151.

20. C.M. Chewar and D.S. McCrickard, Links for a human-centered science of design: Integrated design knowledge environments for a software development process, Proceedings of the Hawaii International Conference on System Sciences, Big Island, HI, United States, 2005, pp. 256.

21. C.W. Choo, The Knowing Organization: How Organizations Use Information to Construct Meaning, Create Knowledge, and Make Decisions, Oxford University Press, 1998

22. R. Conradi and T. Dybå, An empirical study on the utility of formal routines to transfer knowledge and experience, Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Association for Computing Machinery, Vienna, Austria, 2001, pp. 268-276.

23. T.H. Davenport and L. Prusak, Working Knowledge: How Organizations Manage What They Know, Harvard Business School Press, 1998

24. R. De Almeida Falbo, L.S.M. Borges, and F.F.R. Valente, Using knowledge management to improve software process performance in a CMM level 3 organization, Proceedings of the Fourth International Conference on Quality Software, IEEE Computer Society, Braunschweig, Germany, 2004, pp. 162-169.

25. K.C. Desouza. Facilitating tacit knowledge exchange. Communications of the ACM. 46(6) (2003) 85-88

26. K.C. Desouza, Y. Awazu, and Y. Wan. Factors governing the consumption of explicit knowledge. Journal of the American Society for Information Science and Technology. 57(1) (2006) 36-43

27. K.C. Desouza, T. Dingsoyr, and Y. Awazu. Experiences with conducting project postmortems: Reports versus stories. Software Process Improvement and Practice. 10(2) (2005) 203-215

28. M. Dierkes, A. Berthoin Antal, J. Child, and I. Nonaka, Handbook of Organizational Learning and Knowledge, Oxford University Press, 2001

29. T. Dingsoyr, H.K. Djarraya, and E. Royrvik. Practical knowledge management tool use in a software consulting company. Communications of the ACM. 48(12) (2005) 96-100

30. T. Dingsoyr and E. Royrvik, An empirical study of an informal knowledge repository in a medium-sized software consulting company, Proceedings of the International Conference on Software Engineering, Portland, OR, United States, 2003, pp. 84-92.

31. T. Dingsøyr, Knowledge Management in Medium-Sized Software Consulting Companies, Dr. ing. thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2002

32. T. Dingsøyr and R. Conradi. A survey of case studies of the use of knowledge management in software engineering. International Journal of Software Engineering and Knowledge Engineering. 12(4) (2002) 391-414
33. T. Dingsøyr and G.K. Hanssen, Extending Agile Methods: Postmortem Reviews as Extended Feedback, Proceedings of the 4th International Workshop on Learning Software Organizations, Springer Verlag, Chicago, IL, USA, 2002, pp. 4-12.
34. H.D. Doran, Agile Knowledge Management in Practice, Proceedings of the 6th International Workshop on Learning Software Organizations, Springer Verlag, Banff, Canada, 2004, pp. 137-143.
35. T. Dybå, Enabling Software Process Improvement: An Investigation on the Importance of Organizational Issues, Dr. ing thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2001
36. T. Dybå, T. Dingsøyr, and G.K. Hanssen, Applying Systematic Reviews to Diverse Study Types: An Experience Report, Proceedings of the ESEM, Madrid, Spain, 2007,
37. T. Dybå, B.A. Kitchenham, and M. Jørgensen. Evidence-Based Software Engineering for Practitioners. IEEE Software. 22(1) (2005) 58-65
38. M. Earl. Knowledge Management Strategies: Towards a Taxonomy. Journal of Management Information Systems. 18(1) (2001) 215-233
39. M. Easterby-Smith and M.A. Lyles, The Blackwell handbook of organizational learning and knowledge management, Blackwell Publishing, 2003
40. C. Ebert, J. De Man, and F. Schelenz, e-R&D: Effectively Managing and Using R&D Knowledge, in: A. Aurum, et al. (Eds.), Managing Software Engineering Knowledge, Springer-Verlag, 2003, pp. 339-359.
41. J.S. Edwards, Managing Software Engineers and Their Knowledge, in: A. Aurum, et al. (Eds.), Managing Software Engineering Knowledge, Springer-Verlag, Berlin, 2003, pp. 5-27.
42. F.F. Fajtak, Kick-off Workshops and Project Retrospectives: A good learning software organization practice, Proceedings of the 7th International Workshop on Learning Software Organizations, Springer Verlag, Kaiserslautern, Germany, 2005, pp. 76-81.
43. P. Feher and A. Gabor. The role of knowledge management supporters in software development companies. Software Process Improvement and Practice. 11(3) (2006) 251-260
44. R.L. Feldmann and K.-D. Althoff, On the Status of Learning Software Organisations in the Year 2001, Proceedings of the Learning Software Organizations Workshop, Springer Verlag, Kaiserslautern, Germany, 2001, pp. 2-6.
45. H. Folkestad, E. Pilskog, and B. Tessem, Effects of Software Process in Organization Development – A Case Study, Proceedings of the 6th International Workshop on Learning Software Organizations, Springer Verlag, Banff, Canada, 2004, pp. 153-164.
46. R.L. Glass, V. Ramesh, and V. Iris. An Analysis of Research in Computing Disciplines. Communications of the ACM. 47(6) (2004) 89-94

47. G. Grabher and O. Ibert. Bad company? The ambiguity of personal knowledge networks. Journal of Economic Geography. 6(3) (2006) 251-271
48. B.H. Hansen and K. Kautz, Knowledge mapping: A technique for identifying knowledge flows in software organisations, in: Lecture Notes in Computer Science 3281, 2004, pp. 126-137.
49. M.T. Hansen, N. Nohria, and T. Tierney. What is your strategy for managing knowledge? Harvard Business Review. 77(2) (1999) 106 - 116
50. F. Houdek and C. Bunse, Transferring Experience - A practical Approach and its Application on Software Inspections, Proceedings of the 1st Workshop on Learning Software Organizations, Kaiserslautern, Germany, 1999, pp. 59-68.
51. F. Houdek, K. Schneider, and E. Wieser, Establishing Experience Factories at Daimler-Benz. An Experience Report, Proceedings of the 20th International Conference on Software Engineering, Kyoto, Japan, 1998, pp. 443-447.
52. P. Jalote, Knowledge Infrastructure for Project Management, in: A. Aurum, et al. (Eds.), Managing Software Engineering Knowledge, Springer-Verlag, 2003, pp. 361-375.
53. C. Johannson, P. Hall, and M. Coquard, Talk to Paula and Peter - They are Experienced, Proceedings of the 1st Workshop on Learning Software Organizations, Kaiserslautern, Germany, 1999, pp. 69-76.
54. T. Kahkonen, Agile methods for large organizations - Building communities of practice, Proceedings of the Agile Development Conference, IEEE Computer Society, Salt Lake City, UT, United States, 2004, pp. 2-10.
55. K. Kautz and K. Thaysen. Knowledge, learning and IT support in a small software company. Journal of Knowledge Management. 5(4) (2001) 349-357
56. P. Kess and H. Haapasalo. Knowledge creation through a project review process in software production. International Journal of Production Economics. 80(1) (2002) 49-55
57. P. Kettunen. Managing embedded software project team knowledge. IEE Software. 150(6) (2003) 359-366
58. B.A. Kitchenham, *Procedures for Performing Systematic Reviews*, in *Technical Report TR/SE-0401*. 2004, Keele University.
59. B.A. Kitchenham, T. Dybå, and M. Jørgensen, Evidence-Based Software Engineering, Proceedings of the International Conference on Software Engineering, 2004, pp. 273-281.
60. S. Koenig, Integrated process and knowledge management for product definition, development and delivery, Proceedings of, 2003, pp. 133-141.
61. A. Koennecker, J. Ross, and L. Graham, Lessons Learned From the Failure of an Experience Base Initiative Using a Bottom-Up Development Paradigm, Proceedings of the 24th Annual NASA Software Engineering Workshop, Washington, USA, 1999,
62. D. Kolb, Experiental Learning: Experience as the Source of Learning and Development, Prentice Hall, 1984
63. D. Kolb, Management and the learning process, in: K. Starkey (Eds.), How Organizations Learn, Thomson Business Press, London, 1996, pp. 270-287.
64. F. Kurniawati and R. Jeffery, The long-term effects of an EPG/ER in a small software organisation, Proceedings of the Australian Software Engineering Conference, Melbourne, Vic., Australia, 2004, pp. 128-136.

65. D. Landes, K. Schneider, and F. Houdek. Organizational learning and experience documentation in industrial software projects. International Journal of Human Computer Studies. 51(3) (1999) 643-661

66. S.-h. Liao. Knowledge management technologies and applications - literature review from 1995 to 2002. Expert Systems with Applications. 25 (2003) 155-164

67. J. Liebowitz. A look at NASA Goddard space flight center's knowledge management initiatives. IEEE Software. 19(3) (2002) 40-42

68. M. Lindvall, P. Costa, and R. Tesoriero, Lessons Learned about Structuring and Describing Experience for Three Experience Bases, Proceedings of the 3rd International Workshop on Learning Software Organizations, Springer Verlag, Kaiserslautern, Germany, 2001, pp. 106-119.

69. M. Lindvall and I. Rus. Knowledge Management in Software Engineering. IEEE Software. 19(3) (2002) 26 - 38

70. M. Lindvall and I. Rus, Knowledge Management for Software Organizations, in: A. Aybüke, et al. (Eds.), Managing Software Engineering Knowledge, Springer Verlag Berlin, 2003, pp. 73-94.

71. M. Lindvall and I. Rus, Lessons Learned from Implementing Experience Factories in Software Organizations, Proceedings of the 5th International Workshop on Learning Software Organizations, Bonner Köllen Verlag, Luzern, Switzerland, 2003, pp. 59-64.

72. M. Lindvall, I. Rus, R. Jammalamadaka, and R. Thakker, *Software Tools for Knowledge Management*, in *tech. report*. 2001, DoD Data Analysis Center for Software, Rome, N.Y.

73. M. Markkula, Knowledge Management in Software Engineering Projects, Proceedings of the International Conference on Software Engineering and Knowledge Engineering, Kaiserslautern, Germany, 1999, pp. 20-27.

74. N. Martin-Vivaldi, P. Collier, and S. Kipling, Peer Performance Coaching: Accelerating Organizational Improvement through Individual Improvement, Proceedings of the 2nd Workshop on Learning Software Organizations, Oulu, Finland, 2000, pp. 103-112.

75. L. Mathiassen and L. Vogelsang, The role of networks and networking in bringing software methods to practice, Proceedings of the Hawaii International Conference on System Sciences, Big Island, HI, United States, 2005, pp. 256.

76. G. Melnik and F. Maurer, Direct verbal communication as a catalyst of agile knowledge sharing, Proceedings of the Agile Development Conference, Salt Lake City, UT, United States, 2004, pp. 21-31.

77. K. Mohan and B. Ramesh, Managing variability with traceability in product and service families, Proceedings of, 2002, pp. 1309-1317.

78. E. Niemela, J. Kalaoja, and P. Lago. Toward an architectural knowledge base for wireless service engineering. Ieee Transactions on Software Engineering. 31(5) (2005) 361-379

79. I. Nonaka and H. Takeuchi, The Knowledge-Creating Company, Oxford University Press, 1995

80. J. Nørbjerg, T. Elisberg, and J. Pries-Heje, Experiences from using knowledge networks for sustaining Software Process Improvement, Proceedings of the 8th

International Workshop on Learning Software Organizations, Rio de Janeiro, Brazil, 2006, pp. 9-17.

81. T.J. Ostrand and E.J. Weyuker, A Learning Environment for Software Testers at AT&T, Proceedings of the 2nd Workshop on Learning Software Organizations, Oulu, Finland, 2000, pp. 47-54.

82. M.C. Paulk, C.V. Weber, and B. Curtis, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Reading, MA, USA, 1995

83. M. Polanyi, The Tacit Dimension, Doubleday, 1967

84. S. Ramasubramanian and G. Jagadeesan. Knowledge management at infosys. Ieee Software. 19(3) (2002) 53-+

85. E. Ras, G. Avram, P. Waterson, and S. Weibelzahl. Using weblogs for knowledge sharing and learning in information spaces. Journal of Universal Computer Science. 11(3) (2005) 394-409

86. T. Ravichandran and A. Rai. Structural analysis of the impact of knowledge creation and knowledge embedding on software process capability. Ieee Transactions on Engineering Management. 50(3) (2003) 270-284

87. O.M. Rodriguez, A.I. Martinez, A. Vizcaino, J. Favela, and M. Piattini, Identifying knowledge management needs in software maintenance groups: A qualitative approach, Proceedings of the Fifth Mexican International Conference in Computer Science, Colima, Mexico, 2004, pp. 72-79.

88. T.R. Roth-Berghofer, Learning from HOMER, a Case-Based Help Desk Support System, Proceedings of the 6th International Workshop on Learning Software Organizations, Springer Verlag, Banff, Canada, 2004, pp. 88-97.

89. I. Rus, M. Lindvall, and S.S. Sinha, *Knowledge Management in Software Engineering*, in *tech. report*. 2001, DoD Data Analysis Center for Software, Rome.

90. O. Salo, Systematical Validation of Learning in Agile Software Development Environment, Proceedings of the 7th International Workshop on Learning Software Organizations, Springer Verlag, Kaiserslautern, Germany, 2005, pp. 106-110.

91. K. Schneider. What to expect from software experience exploitation. Journal of Universal Computer Science. 8(6) (2002) 570-580

92. K. Schneider, J.-P. Von Hunnius, and V.R. Basili. Experience in implementing a learning software organization. IEEE Software. 19(3) (2002) 46-49

93. J.-P.v.H. Schneider Kurt. Experience reports: process and tools: Effective experience repositories for software engineering (2003

94. D.G. Schwartz, Encyclopedia of Knowledge Management, Idea Group Reference, 2006

95. L. Scott and T. Stålhane, Experience Repositories and the Postmortem, Proceedings of the 5th International Workshop on Learning Software Organizations, Bonner Köllen Verlag, Luzern, Switzerland, 2003, pp. 79-82.

96. J. Segal, Organisational learning and software process improvement: a case study, Proceedings of the 3rd International Workshop on Learning Software Organizations, Springer Verlag, Kaiserslautern, Germany, 2001, pp. 68-82.

97. P.M. Senge, The Fifth Discipline: The Art & Practise of The Learning Organisation, Century Business, 1990

98. D. Skuce. Knowledge Management in Software-Design - a Tool and a Trial. Software Engineering Journal. 10(5) (1995) 183-193
99. R. Stata, Organizational learning: The key to management innovation, in: K. Starkey (Eds.), How organizations learn, Thomson Business Press, London, 1996, pp. 316 - 334.
100. K.E. Sveiby, The New Organizational Wealth: Managing and Measuring Knowledge-Based Assets, Berret-Koehler Pub, 1997
101. A.H. Torres, N. Anquetil, and K. Oliveira, Pro-active dissemination of Knowledge with Learning Histories, Proceedings of the 8th International Workshop on Learning Software Organizations, Rio de Janeiro, Brazil, 2006, pp. 19-27.
102. R. Trittmann, The organic and the mechanistic form of managing knowledge in software development, Proceedings of the 3rd International Workshop on Learning Software Organizations, Springer Verlag, Kaiserslautern, Germany, 2001, pp. 22-26.
103. J.-W. van Aalst, Knowledge Management in Courseware Development, PhD thesis, Technical University Delft, 2001
104. C.G.v. Wangenheim, S. Weber, J.C.R. Hauck, and G. Trentin. Experiences on establishing software processes in small companies. Information and Software Technology. 48(9) (2006) 890-900
105. J. Ward and A. Aurum, Knowledge management in software engineering - Describing the process, Proceedings of the Australian Software Engineering Conference, Melbourne, Vic., Australia, 2004, pp. 137-146.
106. E. Wenger, Communities of practice : learning, meaning and identity, Cambridge University Press, 1998
107. K.P. Yglesias, IBM's reuse programs: Knowledge management and software reuse, Proceedings of the International Conference on Software Reuse, 1998, pp. 156-164.

Appendix

Table 18: Categorized articles, extended

	Systems	Cartographic	Engineering	Commercial	Organizational	Spatial	Strategic
Emp	[11, 17, 20, 30, 64, 98]	[29]	[2, 6, 14, 16, 22, 27, 45, 48, 76, 90, 96, 104]		[47, 75, 80]		[5, 43, 102]
LL	[4, 12, 19, 24, 50-52, 60, 61, 65, 68, 71, 73, 77, 78, 85, 88, 91, 93, 95]		[4, 33, 42, 56, 57, 74, 87, 95, 101]		[53, 54]	[25]	[18, 34, 40, 55, 67, 81, 84, 92, 107]

Table 19: Overview of research methods, extended

Research Method	KM/SE
Action Research	[5, 16, 17]
Case study	[2, 6, 14, 20, 27, 45, 47, 64, 75, 80, 90, 96, 98, 104]
Etnography	[29, 30]
Laboratory Experiment	[76]
Field Study	[11, 22, 43, 48, 102]

Appendix B: Secondary papers

In this appendix we have included the abstract of the seven papers that we have contributed towards, but fell outside the final scope of the thesis:

- SP1: Empirical Study on COTS Component Classification
- SP2: An Empirical Study of COTS Component Selection Processes in Norwegian IT companies
- SP3: An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry
- SP4: Using Open Space Technology as a Method to Harvest Domain Knowledge
- SP5: Future studies of Learning Software Organizations (Combination of SP4 with other papers from LSO’05)
- SP6: Using Rational Unified Process in an SME – A Case Study
- SP7: An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry (SP3 edited for journal)

SP1: Empirical Study on COTS Component Classification

Jingyue Li, Finn Olav Bjørnson, Reidar Conradi

Department of Computer and Information Science,
Norwegian University of Science and Technology, Trondheim, No-7491, Norway
{jingyue, bjornson, conradi}@idi.ntnu.no

Published: Proc. International Workshop on COTS Terminology and Concepts, Redondo Beach, USA, 2004, 4p

Abstract: COTS-based development is gaining more and more attention. Effective COTS component classification will help integrators to successfully control the development process, such as selection and integration. In this paper, we present an empirical study to investigate the characterized classification proposed by previous research. From the result of this study, we conclude that some attributes are not good because they are either not measurable or unnecessary. We also propose one other attribute that will affect the development process dramatically. Our future study will focus on investigating these attributes in a larger sample.

SP2: An Empirical Study of COTS Component Selection Processes in Norwegian IT companies

Jingyue Li[1], Finn Olav Bjørnson[1], Reidar Conradi[1], Vigdis By Kampenes[2]

[1]Dept. of Computer and Information Science
Norwegian Univ. of Science and Technology
NO-7491 Trondheim, Norway
{jingyue,bjornson,conradi}@idi.ntnu.no

[2]Simula Research Laboratory
P.O.BOX 134, NO-1325 Lysaker, Norway
{vigdis@simula.no}

Published: Proc. Of the International Workshop on Models and Processes for the Evaluation of COTS Components (MPEC), Edinburgh, Scotland, 2004, 27-30

Abstract: The use of Commercial-Off-The-Shelf (COTS) software has become more and more important in software development. In COTS-based development, COTS component selection is the most crucial phase. Although some selection processes have been proposed, empirical studies are necessary to assess these processes. This paper describes an exploratory study by structured interviews of 16 COTS-based development projects in Norwegian IT companies. The results indicate that successful COTS component selection can be implemented without using formal processes, and projects with different contexts may use different selection processes. If members in new project has enough practical experience with actual COTS components, such experience can be the dominant factor in selection. In the case of using a new COTS component in the project, hands-on experimentation is needed as an effective way of evaluating the component.

SP3: An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry

Jingyue Li[1], Finn Olav Bjørnson[1], Reidar Conradi[1], Vigdis By Kampenes[2]

[1]Dept. of Computer and Information Science
Norwegian Univ. of Science and Technology
NO-7491 Trondheim, Norway
{jingyue,bjornson,conradi}@idi.ntnu.no

[2]Simula Research Laboratory
P.O.BOX 134, NO-1325 Lysaker, Norway
{vigdis}@simula.no

Published: Proc. the 10th IEEE International Metrics Symposium (Metrics), Chicago, USA, 2004, 72-83

Abstract: More and more software projects use Commercial-Off-The-Shelf (COTS) components. Although previous studies have proposed specific COTS-based development processes, there are few empirical studies to investigate how to use and customize them to different project contexts. This paper describes an exploratory study of state-of-the-practice of COTS-based development processes. 16 software projects in Norwegian IT companies have been studied by structured interviews. The results are that COTS-specific activities can be successfully incorporated in most traditional development processes (such as waterfall or prototyping), given proper guidelines to reduce risks and provide specific assistance. We have identified four COTS-specific activities - the build vs. buy decision, COTS component selection, learning and understanding COTS components, and COTS component integration - and one new role, that of a knowledge keeper. We have also found a special COTS component selection activity for unfamiliar components, combining Internet searches with hands-on trials. The process guidelines are expressed as scenarios and lessons learned, and can be used to customize the actual development processes, e.g. in which lifecycle phase to put the new activities. Such customization crucially depends on project context, such as previous familiarity with possible COTS components and flexibility of requirements.

SP4: Using Open Space Technology as a Method to Harvest Domain Knowledge

Torgeir Dingsøyr[1], Finn Olav Bjørnsson[2]

[1]SINTEF Information and Communication Technology
NO-7465 Trondheim, Norway

[2]Dept. of Information and Communication Systems,
Norwegian University of Science and Technology
NO-7491 Trondheim, Norway

Published: Proc. 7th International Workshop on Learning Software Organizations (LSO), Kaiserslautern, Germany, 2005, 102-106

Abstract: Domain knowledge is a crucial ingredient for companies developing software, yet little attention is paid on how to gain such knowledge in the software engineering literature. We here propose a study on using one large-group intervention technique – Open Space Technology – to increase the domain knowledge of developers in a software project.

SP5: Future studies of Learning Software Organizations

Kari Smolander[1], Kurt Schneider[2], Torgeir Dingsøyr[3], Finn Olav Bjørnsson[4], Pasi Juvonen[1], Päivi Ovaska[1]

[1]South Carelia Polytechic, Koulukatu 5 B, 55120 Imatra, Finland

[2]Software Engineering Group, Universität Hannover, Welfengarten 1, 30167 Hannnover, Germany

[3]SINTEF Information and Communication Technology, NO-7465 Trondheim, Norway

[4]Dept. of Information and Communication Systems, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway

Published: Professional Knowledge Management, Springer LNAI 3782, 2005, 134-144

Abstract. We suggest to study learning software organizations in three projects; one to analyse the current situation for local software and system houses, one to study improvement and learning through examining knowledge flows, and a third to study the impact of a large-scale interaction process: Open Space Technology to share domain knowledge.

SP6: Using Rational Unified Process in an SME – A Case Study

Geir Kjetil Hanssen[1], Hans Westerheim[1], Finn Olav Bjørnson[2]

[1]SINTEF ICT, N-7465 Trondheim, Norway
{geir.kjetil.hanssen, hans.westerheim}@sintef.no

[2]Norwegian University of Science and Technology, N-7491 Trondheim, Norway
bjornson@idi.ntnu.no

Published: Proc. EuroSPI'05 conference, Budapest, Springer LNCS 3792, 2005, 142-150

Abstract: The Rational Unified Process (RUP) is a comprehensive software development process framework emphasizing use-cases, architecture focus and an iterative approach. RUP is widely known and many organizations have tried to adopt it. Being a framework, RUP has to, in some way, be tailored to the specific context of use, no software development project is alike. This paper presents a case study of a Norwegian SME that tried to adopt RUP in the simplest way, by introducing the methodology by providing comprehensive documentation and some simple training. Our study shows that the use of RUP had some positive effects but also that the use has been scattered. Interviews with users of RUP show that there is a great need of better training and practical support in getting most value out of RUP. The key message is that if you consider taking RUP into use you have to invest resources in it. Training and support are key success factors.

SP7: An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry

Jingyue Li[1], Finn Olav Bjørnson[1], Reidar Conradi[1], Vigdis By Kampenes[2]

[1]Dept. of Computer and Information Science
Norwegian Univ. of Science and Technology
NO-7491 Trondheim, Norway
{jingyue,bjornson,conradi}@idi.ntnu.no

[2]Simula Research Laboratory
P.O.BOX 134, NO-1325 Lysaker, Norway
{vigdis}@simula.no

Published: Journal of Empirical Software Engineering, 11(3), 2006, 433-461

Abstract: More and more software projects use Commercial-Off-The-Shelf (COTS) components. Although previous studies have proposed specific COTS-based development processes, there are few empirical studies that investigate how to use and customize COTS-based development processes for different project contexts. This paper describes an exploratory study of state-of-the-practice of COTS-based development processes. Sixteen software projects in the Norwegian IT companies have been studied by structured interviews. The results are that COTS-specific activities can be successfully incorporated in most traditional development processes (such as waterfall or prototyping), given proper guidelines to reduce risks and provide specific assistance. We have identified four COTS-specific activities – the build vs. buy decision, COTS component selection, learning and understanding COTS components, and COTS component integration – and one new role, that of a knowledge keeper. We have also found a special COTS component selection activity for unfamiliar components, combining Internet searches with hands-on trials. The process guidelines are expressed as scenarios, problems encountered, and ex-amples of good practice. They can be used to customize the actual development processes, such as in which lifecycle phase to put the new activities into. Such customization crucially depends on the project context, such as previous familiarity with possible COTS components and flexibility of requirements.