Knowledge Management in Medium-Sized Software Consulting Companies

An investigation of Intranet-based Knowledge Management Tools for Knowledge Cartography and Knowledge Repositories for Learning Software Organisations

Torgeir Dingsøyr

Knowledge Management in Medium-Sized Software Consulting Companies

An Investigation of Intranet-based Knowledge Management Tools for Knowledge Cartography and Knowledge Repositories for Learning Software Organisations

Submitted for the Partial Fulfillment of the Requirements for the Degree of Doktor Ingeniør

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology
January, 2002

ISBN 82-7477-107-9
ISSN: 1502-1408

Information concerning this publication can be directed to:
Phone: +47 22 85 30 30
Fax: +47 22 85 30 39
E-mail: post@unipub.no
The book can also be purchased at www.gnist.no

Cover: Askim Grafix AS
Printed in Norway by: GCSM AS, Oslo 2002

Layout: Hanne Holmesland

Unipub forlag is a subsidiary company of Akademika AS, owned by Studentsamskipnaden i Oslo

Abstract

Companies that develop software have a pressure from customers to deliver better solutions, and to deliver solutions faster and cheaper. Many researchers have worked with suggestions on how to improve the development process; software process improvement. As software development is a very knowledge intensive task, both researchers and industry have recently turned their attention to knowledge management as a means to improve software development. This often involves developing technical tools, which many companies have spent resources on. But the tools are often not used in practise by developers and managers in the companies, and it is often unknown if the tools improve how knowledge is managed.

In order to build efficient knowledge management tools, we need a better understanding of how the tools that exist are applied and used in software development.

We present and analyse eight case studies of knowledge management initiatives from the literature. We found evidence of improved software quality, reduced development costs and evidence of a better working environment for developers as a result of these initiatives.

Further, we examine success criteria in knowledge management codification initiatives, based on Intranet tools in medium-sized software companies. We found four factors that we consider important: Having a culture for sharing knowledge, having a stable focus on knowledge management, developing knowledge management tools incrementally, and coupling knowledge management initiatives well to business goals. This research was based on participation with software companies in improvement projects.

In addition, we investigate how knowledge management tools are used for different purposes by different groups of users in two software consulting companies. They use tools both as support for personalization and codification strategies. The consulting companies are two medium-sized Norwegian companies with 40 and 150 employees, which work in

development projects that lasts from a few weeks to several years. We used semi-structured interviews with developers, project managers and managers, examined logs of tool usage, and company-internal minutes from development meetings, as well as handbooks, project plans and annual reports.

The frequency of usage varied between the two companies: in one, most employees used tools on a daily basis, whilst in the other, employees used tools weekly. We find that tools for codification are in use for transferring knowledge from projects in order to solve technical problems, get an overview of technical problem areas, avoiding rework in having to explain many people about the same technical solution, improving the employees' work situation by tips on better configuration of technical tools, and also for finding who knows what in the organisation. The tools for personalization are in use for searching for competence to solve technical problems, resource allocation, finding projects and external marketing, and for competence development. In all, we found a variety of uses of a variety of tools by several groups of employees in a company.

«The maturation of the information technology revolution in the 1990s has transformed the work process, introducing new forms of social and technical division of labor.»

Manuel Castells in The Rise of the Network Society

Contents

Acknowledgements

During the work on this thesis, I have benefited from a lot of communication with many people, who have provided inspiration and motivation. First of all, I would like to thank my supervisor, Reidar Conradi, who has commented on an enormous number of drafts through the last four years. Also, thanks to present and former members of the software engineering group at the Department of Computer and Information Science, NTNU, for providing a good work environment: Elisabeth Bayegan, Roxana Diaconescu, Monica Divitini, Ekaterina Prasolova-Førland, Letizia Jaccheri, Jens-Otto Larsen, Øystein Nytrø, Tor Stålhane, Sivert Sørumgård, Carl-Fredrik Sørensen, Marco Torchiano and Alf Inge Wang.

Another source of inspiration has been the possibility to participate in two research projects during my PhD work: The Software Process Improvement for Better Quality (SPIQ) and Process Improvement for IT-industry (PROFIT) projects. I would like to thank participants from Sintef Telecom and Informatics: Tore Dybå, Geir Kjetil Hanssen, Nils Brede Moe and Kari Juul Wedde (now Clustra) as well as from the University of Oslo: Erik Arisholm, Dag Sjøberg, Magne Jørgensen and project manager Tor Ulsund from Bravida Geomatikk. Thanks are also due to the contact persons from companies that participated in these projects, which provided a pragmatic research environment that greatly influenced the focus of this PhD. I am also grateful to the Norwegian Research Council who financed the PhD work.

A part of the work for the thesis was done as fieldwork in two companies, and I am deeply grateful to these companies and the contact persons and the people I interviewed for their willingness to share experience about their knowledge management efforts.

During the last phase of the work, I was able to stay at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern, Germany. Many people deserve thanks for both social and professional inclusion. It was especially nice to be able to work on the COIN Experience Factory project with Björn Decker and Markus Nick, which made me see knowledge management from another perspective. And I am

very grateful to Klaus-Dieter Althoff for organising the stay, and also to the rest of the Systematic Learning and Improvement group at the institute for numerous discussions.

Through the years that I have been working as a doctoral candidate, I have benefited greatly from discussions with students that I have supervised in project and diploma work: Arne Bakkebø, Terje Nygaard, Bjørgulv O. Sandanger, Thies Schrader, Torkel Westgaard, Helge Jenssen and Bjørn-Ovin Wivestad. In the early phase of thesis work, I also had many discussions with Bent Ingebretsen, who had written his masters thesis in this field.

There are also other people that I would like to thank for collaboration during the thesis work: Emil Røyrvik on the Kunne project at Sintef Technology Management for discussions and writings on skills management. Trond Knudsen at Norsk Regnesentral for letting me present an early version of research approach which provoked many new thoughts. Petter Gottschalk at the Norwegian School of Management BI, for giving me an early version of his book on knowledge management which directed me to new references. Also, I would like to thank Frank Maurer at the University of Calgary and Mirjam Minor at Humboldt Universität Berlin for organising guest lectures where some of the material in the thesis was presented and discussed.

I am further grateful to the people who have commented on parts of the thesis: Of course, Reidar Conradi as my supervisor, but also Stefan Biffl at the Technical University of Vienna, Magne Jørgensen at the University of Oslo, and Monica Divitini, Letizia Jaccheri, Roxana Diaconescu and Alf Inge Wang at the Department of Computer and Information Science at NTNU. I am also grateful to Terje Brasethvik at the Information Systems group at NTNU for a number of discussions on research topics and comments on drafts. Also thanks to Gavin Gaudet for tips on how to improve the oral English in the Empirical Investigations chapter. And thanks to Preben Randhol for help with scripts to handle usage logs.

Finally, I would like to thank family and friends for inspiraton and encouragement, and especially Sissel for her patience.

Trondheim, January 21st
Torgeir Dingsøyr

List of Figures

List of Tables

1 Introduction

This thesis is about how Intranet-based Knowledge Management Tools can be used to support what has been called a «Learning Software Organisation». An Intranet-based tool is a software program that provides help for software developers. We will define what we mean by a tool more precisely later.

Software development usually takes place in team-based projects where the participants work towards a shared goal. Many companies have problems with transferring what people learn in one project to other projects in the same company. Knowledge Management is a set of strategies and techniques to increase the transfer and use of different types of knowledge in a company or organisation.

We find many knowledge management tools and methods in companies and in the research literature, but most of the scientific work on tools is concentrating on technology to build such tools; on the structure of knowledge and technical work on retrieval mechanisms. Also, work on knowledge management methods is usually describing an ideal way of collecting and sharing knowledge, which is often difficult to reproduce in practise. There is little work on how tools and methods for knowledge management are actually applied in the software engineering domain. Also, many tools that are introduced in companies are abandoned later. This is often because they turned out not to be so useful as people thought before they were introduced.

We think that we would be able to design better tools and methods, if we knew more about how the existing tools are used - or why they are not used.

In this thesis we discuss how companies can improve their knowledge management by adjusting Intranet-based knowledge management tools, and thus become more of a learning organisation. We will base this discussion on an examination of tools and initiatives that are used in medium-sized companies that develop software. These medium-sized companies are four case companies in a prestudy, and a main and a contrast

case in a main study - as well as reports of knowledge management tools from the literature.

Now, we go on to define a problem outline for this thesis that will be further narrowed later and state the main contributions of this thesis. Then, we briefly state what main choices we have made for carrying out research. Further, we narrow the scope of this work, and finally give an overview of the structure of the thesis.

1.1 Problem Outline

In this thesis, we are interested in studying how tools for knowledge management are used in medium-sized companies that develop software. The specific tools are Intranet-based tools that companies have produced themselves. There are, however, many such tools, and we will only be concerned with Knowledge Repository and Library, and Knowledge Cartography Tools. We will introduce these types later. and argue why these are particularly interesting to examine.

The type of companies where we have studied this phenomenon is in medium-sized companies in Norway that develop software. By medium-sized we will mean companies with from 50 to 500 employees.

Many knowledge management tools are in use in the software industry. But there has been done little work on how these tools actually work in practise. Also, many research prototypes for knowledge management tools exist in the research literature. But not many of them has made it into industrial practise.

We are then asking the following research question:

- How can Intranet-based knowledge management tools be used in medium-sized software consulting companies to facilitate a «learning software organisation»?

This research question will be further discussed and elaborated after we have introduced more theory. We will also elaborate what we mean by a «Learning Software Organisation».

The critical reader might already now ask: But do these knowledge management tools help solve the problems that the software industry has (which will be described in the next chapter)? The answer is: we are not sure. But we think we need to know more about the tools in use, and about how they are used before we can begin to answer the question of whether they are solving problems or not, and of how cost-effective they are.

But is it really any use in studying such tools? The technology is changing so fast. When we have completed this study, the tools will be completely different! Although we think that developing tools for knowledge management is a long process, and the ones we will study are by no means «completed» - we still think it is important to study how they work, before moving on to something else. It has been claimed that it is a general problem in software engineering, that we do not systematically study the effect of technology and methods, before we jump on to newer technologies. We think it is a sound scientific task to analyse the impact of «new» tools. Yet, we acknowledge that the results might be a bit «old» when we finish.

Then, when we examine such tools, what is the relation between their usage and the potential improvement of the productivity or quality of the software that is developed? It is a long chain of events from the effects of a knowledge management tool, to this knowledge being learned and used by employees, which should then finally affect the quality of the developed software or the productivity of the software development team. We do not intend to show a causal relationship between these factors, but we think a knowledge management tool is one of many factors in a good work environment that can stimulate learning, creativity, and employee motivation, which will affect the quality of the output. But we limit ourselves here to study how knowledge management tools can be used, leaving more «hard measurements» for further work.

1.2 Claimed Contributions

The work in this thesis can be divided into four major phases, where we claim to have some contributions in each, related to the field of studying Learning Software Organisations by empirical methods. Some work in

the thesis has been published before, and we give references to these papers for each phase:

- Literature study: We present literature on knowledge management in software engineering, and have made a taxonomy of knowledge management tools based on findings from the literature. We have also surveyed existing case studies of how knowledge management tools are applied in companies that develop software, and present, and discuss these approaches. This work can be found in chapter 3 in the thesis, and in papers 1 and 8.
- Method for experience capture: We have contributed in developing a method to capture experience from completed software projects though a group process: lightweight postmortem reviews. This method is given as an example of experience capture methods in chapter 3, and is described in further detail in papers 2, 5 and 6.
- Four cases studies on knowledge management in software engineering companies: Here, we studied four companies that have applied different knowledge management initiatives, and discuss success factors. The cases are presented in chapter 5.1, and discussed in chapter 6. This analysis has also been published as paper 4.
- Deep case study and a contrast case: We examine further what kind of knowledge management tools that exist in two companies, and describe how different groups of users apply them. The cases are presented in chapter 5.2 and 5.3, and are discussed in chapter 6. Some of the work here on Skills Management has been published in papers 3 and 10.

We have further published paper 7 as a first discussion on the selected research topic and research questions in this thesis, that can be found in chapter 4. Finally, paper 9 gives a further description of knowledge management tools than the ones that can be found in chapter 3.

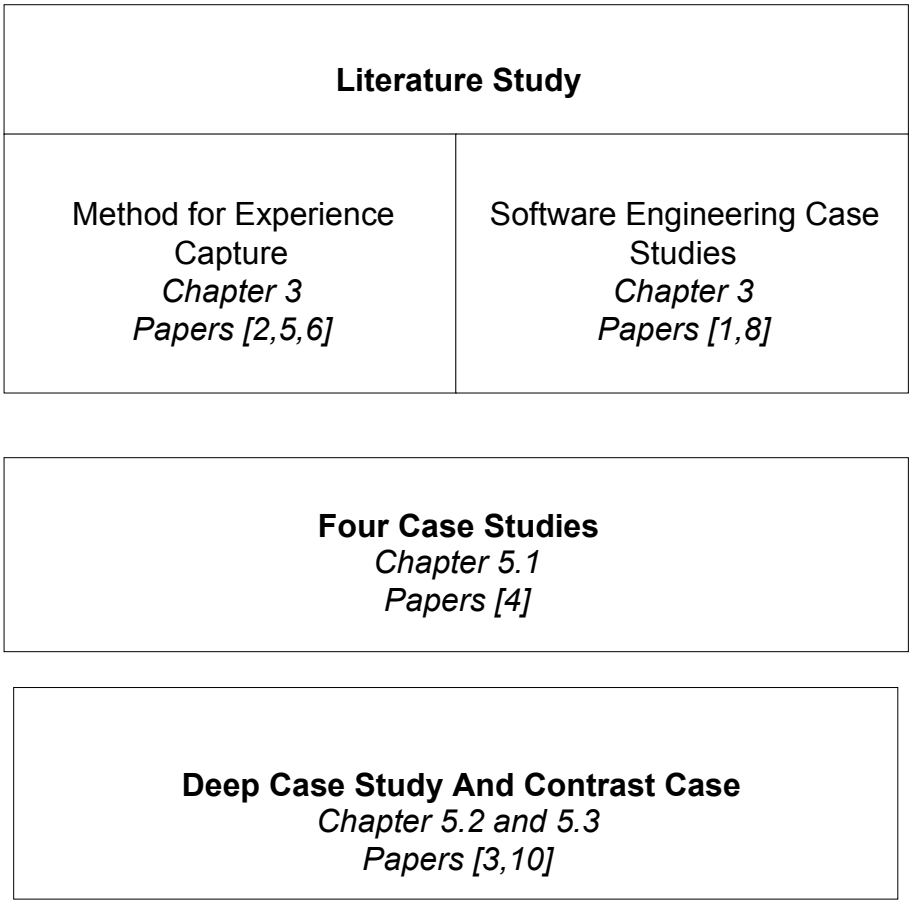


Figure 1.1: The main contributions in this thesis, with references to thesis chapters and published papers.

The following are the papers that has been published or are undergoing a publication process:

Journal articles

1. Dingsøyr, Torgeir, Conradi, Reidar: A Survey of Case Studies of Knowledge Management in Software Engineering, submitted to International Journal of Software Engineering and Knowledge Engineering. A previous version of this paper was published as paper 9.

2. Birk, Andreas, Dingsøyr, Torgeir, Stålhane, Tor: Postmortem: Never leave the project without it, submitted to IEEE Software, special issue on knowledge management in software engineering.

3. Dingsøyr, Torgeir, Djarraya, Hans Karim, Røyrvik, Emil: Managing Hard Skills: Findings from Practical Tool Use in a Software Consulting Company, submitted to IEEE Software, special issue on knowledge management in software engineering.

Book chapter

4. Dingsøyr, Torgeir, Conradi, Reidar: Knowledge Management Systems as a Feedback Mechanism in Software Development Processes: A Search for Success Criteria, submitted as a chapter to a book on Feedback and Evolution in the Software Process. A revised and extended version of: Conradi, Reidar and Dingsøyr, Torgeir (2000) Software experience bases: a consolidated evaluation and status report, Second International Conference on Product Focused Software Process Improvement, PROFES 2000, June 20-22, Oulu, Finland, Springer Verlag, vol. 1840, pp. 391 - 406.

Conference papers

5. Dingsøyr, Torgeir, Moe, Nils Brede and Nytrø, Øystein (2001) Augmenting Experience Reports with Lightweight Postmortem Reviews, Third International Conference on Product Focused Software Process Improvement, 10-13 September, Kaiserslautern, Germany, Springer Verlag, Lecture Notes in Computer Science, vol. 2188, pp. 167 - 181. Also published at the Norwegian Informatics Conference (NIK) 2001, Tromsø.

6. Stålhane, Tor, Dingsøyr, Torgeir, Moe, Nils Brede and Hanssen, Geir Kjetil (2001) Post Mortem - An Assessment of Two Approaches, EuroSPI, 10-12 October, Limerick, Ireland.

Workshop papers

7. Dingsøyr, Torgeir (2000) Focus for planned research: Knowledge Management for Software Process Improvement, The Ninth Nordic

Workshop on Programming Environment Research, 28-30 May, Lillehammer, Norway.

8. Dingsøyr, Torgeir (2000) An evaluation of Research on Experience Factory, Workshop on Learning Software Organisations at the international conference on Product-Focused Software Process Improvement, Oulo, Finland, University of Oulu, VTT Electronics, Fraunhofer IESE, pp. 55 - 66.

9. Dingsøyr, Torgeir (2000) An Analysis of Process Support in Knowledge Management Tools for Software Engineering, Workshop on Flexible Strategies for Maintaining Knowledge Containers,14th European Conference on Artificial Intelligence, 20-25. August, Berlin, Germany, Humboldt-Universität zu Berlin, ECAI Workshop Notes, pp. 6 - 13.

10. Dingsøyr, Torgeir and Røyrvik, Emil (2001) Skills Management as Knowledge Technology in a Software Consultancy Company, Learning Software Organizations Workshop, 12 - 13 September, Kaiserslautern, Germany, Springer Verlag, Lecture Notes in Computer Science, vol. 2176, pp. 96-107.

1.3 Chosen Research Strategy

In researching the question outlined in section 1.1, we have chosen to investigate it in a real environment. That is, to go into a real organisation, and study tools in «vivo». We will discuss this further in the Research Methods and Design chapter. The main reasons for choosing to study real organisations, and doing case and field studies, are that:

- Many prototype knowledge management tools are already developed in research institutions, so the need for making more prototypes is small.
- Few studies exist on how knowledge management tools are used in software companies.

In software engineering, several environments have expressed the need for a more empirical basis of software engineering, promoting what has been called *empirical software engineering*.

In empirical software engineering, it is necessary to use different research methods than normally applied in software engineering. This is because we have no strict control of the environment. Also, in our case, there is relatively little information to find about the usage of knowledge management systems in the research literature.

In studying organisations, we have used research methods that are common in social science, but not in technology-oriented disciplines such as software engineering. A common problem when using such methods is that: «technologists regard sociologists as, apparently, merely wishing to observe and give an account of what they observe, with no interest necessarily in this leading to social action». While on the other hand, «sociologists regard technologists as simply wanting plans of action to make their technology more 'effective'» (Low et al., 1996). Here, we hope that our proposed theory will be seen as a contribution to better understand the tools, and then be useful for anyone wanting to improve the design or usage of such tools later.

Much of the work in software engineering has been done in the spirit of modernity; with a rational view that the problems at hand can be solved if we just establish good enough work methods and tools. The search for a silver bullet (which will be discussed further in the next chapter) is evidence of such a view.

However, many people now have a more post-modern view of software development. That is, it is futile to «solve problems» related to organizational, human and technological factors by say technology alone. Instead of looking for a silver bullet, we can only hope to find a set of «weapons» - that will help us to reduce the impact of some problems as they appear to some people.

In fact, the whole idea of software «engineering» is questioned by some environments (see an interesting discussion on the engineering metaphor in (Bryant, 2000)). Engineering is often associated with words like «science», «mathematics», as well as «practical methods». But we could also see software development as a creative task (Glass, 1995), where for example improvisation (Dybå, 2000) is more important than rigour.

In this thesis we will adopt a subjectivist, or postmodern view, that different people might have different goals, and they do not necessarily

always act in a pre-planned or even rational manner. In studying how people use knowledge management tools, we consider the software practitioners (or «community of practise») to be the true, skilled, experts to judge what kind of tools they find useful or not. Therefore, we have opted for a research strategy with a close interaction with developers, project managers and management in the field. We will discuss this further in our chapter about research goals, method and design.

1.4 Research Context

The work which was performed in this thesis was a part of two larger research projects on software process improvement (SPI) which involved many Norwegian companies that develop software.

The Software Process Improvement for Better Quality (SPIQ) project aimed to increase the competitiveness of 12 participating Norwegian software companies, by creating an improvement environment in the companies, and introducing ideas from an American context, like the Experience Factory, and adopting it for small and medium-sized enterprises in Norway (Conradi, 1996). It also included pilot projects for improvement in companies, as well as discussion forums for issues related to process improvement. Further, it contained dissemination activities like conferences and the writing of a method handbook for process improvement in Norwegian (Dybå et al., 2000). This project lasted from 1997 to 1999.

This project was followed by the Process Improvement for IT industry (PROFIT) project, which focused more on software process improvement in companies with frequent changes in technology and market. Can such companies benefit from the same improvement initiatives as more stable organisations? This was one of the major questions in this project, which is still ongoing, and involved eight companies from the start. This project lasts from 2000 to 2002.

1.5 Scope

In this thesis, we are concerned with how Intranet-based tools are used for knowledge management in medium-sized organisations that develop

software. We have thus limited the field of knowledge management to those processes that can be supported by computer tools, and specifically tools with a web-interface on a company-internal Intranet. We also concentrate on a specific set of tools that will be discussed later. Further, we have limited the usage of these tools to the domain of software development and maintenance, and specifically in medium-sized companies, where most of the development is done «in-house», and where most of the staff spends much of their working day in front of a computer.

When we examined the knowledge management tools, we have only looked at how they are used. We have not looked at issued in developing such tools, and not on economical issues - whether they are cost-effective or not.

We have neither looked at specific tools for reusing code or other software artifacts, but at tools that operate on a higher level of abstraction. But these tools may be linked to code, like a system that help you solve problems by showing example code. In the companies where we have been working, reuse of code is usually organised through development of software libraries of «baseline products» that get input from all people in the organisation.

To introduce knowledge management as an «improvement» in an organisation is of course not without problems. What some people in the organisation see as «improvements» might be seen as «deteriorating» efforts by other people. For example, some employees might think that their knowledge is ignored by a company, because it is not included in a computer tool. This, and other political issues in deploying knowledge management tools are not issues that we will discuss here.

1.6 Structure of the Thesis

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

Chapter 2: Software Development; Problems and Remedies. In this chapter, we discuss what software development is about, and some of the challenges the field is concerned with. We also discuss some of the main improvement initiatives that have been in the field, and discuss one of them,

namely knowledge management and learning organisations in more detail. Finally, we give an overview of research methods in software engineering.

Chapter 3: Knowledge Management: In General and in Software Engineering. Here we first discuss knowledge management in general, and then specifically its application in software engineering. We discuss terms like experience, information and knowledge, and other common terms in the knowledge management field, like organisational memory, corporate memory, and experience factory. We also examine how knowledge is transferred in an organisation, and introduce a knowledge management program as a strategy, a set of processes and a set of tools. We present case studies on knowledge management tools in companies that develop software, found in the literature.

Chapter 4: Research Goals, Method and Design. Here we further specify our research goals, using concepts from chapter 3. We list the topics of interest in the form of research questions. We present the research method that we selected, with arguments for why this approach is suiting the topics under study.

Chapter 5: Empirical Investigation. First, we present a prestudy of four case studies of knowledge management programs in Norwegian companies. Then, we present two companies where we did case studies, together with projects that we followed in each of them. We present the infrastructure for knowledge management that exist in the companies, and our findings on the usage of them.

Chapter 6: Discussion and Analysis. We discuss the findings from the literature, our prestudy and main study cases in light of the theory which is given in chapter 3.

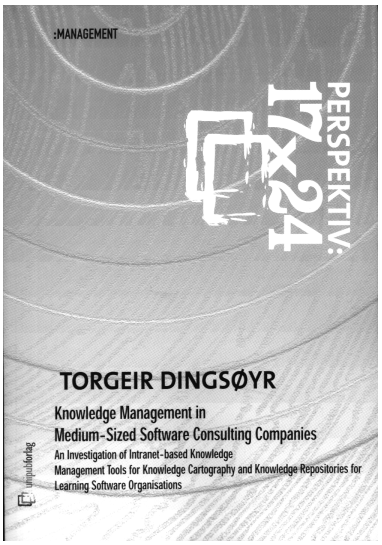
Chapter 7: Conclusion and Further Work. We sum up the main findings from the discussion, and outline possible further work in the field of learning software organisations.

Appendix A: Interview guides - here we present the interview guides that was used in semi-structured interviews in the two companies in the main study.

Appendix B: Processed Usage Logs - Here, we list processed usage data from Knowledge Management Tools in Alpha, one of the main study companies.

This is a doctoral thesis, written for the research community. It is not the intention to come up with direct, practical aid for companies on how to improve their knowledge management, but more to bring forward theory about how knowledge management is used. This will hopefully make it into practise, but it is out of the scope to concentrate on that issue. That is the responsibility of the research field as a whole.

Reading this thesis requires knowledge of software engineering and specifically software process improvement, and what has been called learning software organisations (knowledge management in software engineering). It also requires knowledge of research methods in general.



newrelease!

TORGEIR DINGSØYR
Knowledge Management in Medium-Sized Software Consulting Companies
ISBN: 82-7477-107-9
PRICE: 278,- NOK
PAGES: 206

Torgeir Dingsøyr is a research scientist at the SINTEF Telecom and Informatics research foundation in Trondheim, Norway. He wrote this doctoral thesis at the Department of Computer and Information Science, Norwegian University of Science and Technology.

For ordering and information about the book or its author, contact us by
phone: +47 22 85 33 00
fax: +47 22 85 30 39
email: post@unipub.no
or by mail
Unipub AS, Postboks 84 Blindern
N-0314 Oslo
NORWAY

Software companies are under pressure from customers to deliver solutions faster, cheaper, and with higher quality. As software development is a very knowledge intensive task, both researchers and industry have recently turned their attention to knowledge management as a means to improve software development. This often involves developing technical tools. But the tools are often not used in practise by developers and managers in the companies, and it is often unknown if the tools improve how knowledge is managed.

In order to build efficient knowledge management tools, we need a better understanding of how the tools that exist are applied and used in software development.
This doctoral thesis examines how software consulting companies use Intranet tools to share knowledge across project - to improve software development.
The thesis contains depth studies of how two organisations work with knowledge management, and explains how knowledge management tools are used in different ways, and with different frequencies. The usage varies after what work tasks employees have, and also after their personal taste.

Torgeir Dingsøyr is a research scientist at the SINTEF Telecom and Informatics research foundation in Trondheim, Norway. He wrote this doctoral thesis at the Department of Computer and Information Science, Norwegian University of Science and Technology. He has published papers on knowledge management in software engineering, case-based reasoning and on software engineering education. In 2001, he spent half a year as a guest researcher at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern, Germany.

PLEASE SEND ME ___ COPY(S) OF
Knowledge Management in Medium-Sized Software Consulting Companies
Company:_____
Name:_____
Billingaddress_____
Zip:_____
City:_____
Country:_____
Signature:_____

Selected papers from the thesis

Article:

Andreas Birk, Torgeir Dingsøyr, and Tor Stålhane, “Postmortem: Never leave a project without it,” *IEEE Software, special issue on knowledge management in software engineering*, no. 3, vol. 19, pp. 43 - 45, 2002.

Postmortem: Never Leave a Project without It

Andreas Birk, *sd&m*
Torgeir Dingsøyr, *Sintef Telecom and Informatics*
Tor Stålhane, *Norwegian University of Science and Technology*

In every software project, the team members gain new knowledge and experience that can benefit future projects and each member's own professional development. Unfortunately, much of this knowledge remains unnoticed and is never shared between individuals or teams.

Our experience with project *postmortem analysis* proves that it is an excellent method for knowledge management,[1] which captures experience and improvement suggestions from completed projects and works even in small- and medium- size companies that cannot afford extensive KM investments. However, PMA has been mainly advocated for situations such as completion of large projects, learning from success, or recovering from failure.[2–4]

When used appropriately, PMA ensures that team members recognize and remember what they learned during a project. Individuals share their experiences with the team and communicate them to other project groups. Additionally, PMA identifies improvement opportunities and provides a means to initiate sustained change.

We have applied a lightweight approach to PMA in several projects[5,6] by focusing on a few vital principles:

- PMA should be open for participation from the entire team and other project stakeholders.
- Goals can—but need not—provide a focus for analysis.
- The PMA process comprises three phases: preparation, data collection, and analysis. For each phase, team members can apply a number of fairly simple methods, such as the KJ method (after Japanese ethnologist Jiro Kawakita)[7] that collects and structures the data from a group of people.

Although primarily used for large projects and companies, postmortem analysis also offers a quick and simple way to initiate knowledge management in small- or medium-size software projects.

Preparation

When we conduct PMA in software companies, two software process improvement group members work as facilitators together with two to all project team members. Facilitators organize the analysis, steer the discussion, and document the results. They can be employees in the company where the PMA is conducted or external, as we are. External facilitators often have an advantage performing the PMA because participants regard them as more neutral and objective. However, they might not know the company as well as internal facilitators, so preparation is important.

During the preparation phase, we walk through the project history to better understand what has happened. We review all available documents, such as the work breakdown structure, project plans, review reports, and project reports.

We also determine a goal for the PMA.

0740-7459/02/$17.00 © 2002 IEEE

Once the group identifies the important topics, we must prioritize them before proceeding with the analysis.

Goals might be "Identify major project achievements and further improvement opportunities" or "Develop recommendations for better schedule adherence." If a PMA does not have a specific focus to guide our preparation, we briefly discuss the project with the project manager and key engineers.

We find it practical to distinguish between two PMA types: One is a general PMA that collects all available experience from an activity. The other is a focused PMA for understanding and improving a project's specific activity, such as cost estimation. It helps to explicitly state goals for both of these PMA variants during this phase.

Data collection

In the data collection phase, we gather the relevant project experience. Usually, project team members and stakeholders have a group discussion, or experience-gathering session. We can often conduct data collection and the subsequent analysis within the same session. You shouldn't limit experience gathering to the project's negative aspects, such as things to avoid in the future. Instead, maintain a balance by identifying a project's successful aspects, such as recommended practices. For example, during a PMA at a medical software company, the team realized that the new incremental software integration process significantly improved process control and product quality. Integration had been so smooth that without the PMA, its important role might have gone unnoticed.

Some techniques that we find useful for data collection include

- *Semistructured interviews*. The facilitator prepares a list of questions, such as "What characterizes the work packages that you estimated correctly?" and "Why did we get so many changes to the work in package X?"
- *Facilitated group discussions*. The facilitator leads and focuses the discussion while documenting the main results on a whiteboard.
- *KJ sessions*. The participants write down up to four positive and negative project experiences on post-it notes. Then they present their issues and put the notes on a whiteboard. The participants rearrange all notes into groups according to topic and discuss them.

Once the group identifies the important topics, we must prioritize them before proceeding with the analysis. This will ensure that we address the most significant issues first.

For example, during a PMA we performed in a satellite software company, frequent and late requirements changes emerged as an important topic. A software developer commented that during the project, team members found it difficult to identify when the requirements had changed, so much so that the code had to be rewritten completely. In such situations, they made a few wrong decisions, which reduced the software's quality. After this PMA session, other project members made requirements changes a high-priority topic for analysis.

Analysis

In this phase, as facilitators, we conduct a feedback session in which we ask the PMA participants: "Have we understood what you told us, and do we have all the relevant facts?"

When we know that we have sufficient and reliable data, we use Ishikawa diagrams[6] in a collaborative process to find the causes for positive and negative experiences. We draw an arrow on a whiteboard, which we label with an experience. Then, we add arrows with causes—which creates a diagram looking like a fishbone. In our example from the satellite software company, we found four causes for changing requirements: poor customer requirements specification, new requirements emerging during the project, little contact between the customer and software company, and the software company's poor management of requirements documents.

Because PMA participants are a project's real experts and we have time limitations, we perform all analysis in this step.

Results and experience

Facilitators document the PMA results in a project experience report. The report contains

- A project description, including products developed, development methods used, and time and effort needed
- The project's main problems, with descriptions and Ishikawa diagrams to show causes
- The project's main successes, with descriptions and Ishikawa diagrams

- A PMA meeting transcript as an appendix, to let readers see how the team discussed problems and successes

In an example from the satellite software company, facilitators wrote a 15-page report in which they documented the problem with changing requirements with an Ishikawa diagram that showed the four main causes. After facilitators submit a report, the knowledge management or quality department must follow up.

In our experience, PMA is suitable when a project reaches a milestone and when the company is looking for qualitative experience that will help improve a similar, future project. You should not apply PMA in situations with unfinished activities, or when serious underlying conflicts might remove the focus from improvement. If the atmosphere isn't appropriate for discussing a project's problems, we prefer using approaches other than PMA, such as those outlined in *Project Retrospectives: A Handbook for Team Reviews*.[2] When there have been serious conflicts in the project, this is more appropriate for managing the risk that discussions degenerate into a hunt for scapegoats. Finally, you must have enough time for following up on PMA results.

In our experience, if teams apply PMA in the right setting, it is an excellent step into continuous knowledge management and improvement activities. It makes project team members share and understand one another's perspectives, integrates individual and team learning, and illuminates hidden conflicts. It documents good practice and problems, and finally, it increases job satisfaction by giving people feedback about their work.

Performing a PMA can even improve project cost estimation. We applied PMA to three projects in an Internet software development company, which all had serious cost overruns. The company could not allocate workers with skills specific to the project. This led to a need for courses—the team's experts had to act as tutors for the rest of the team and were distracted from their roles in the project. By performing the PMA, the company realized the gravity of the qualification issue and how it led to the project going over budget. As an improvement action, a training budget was set up on the company level instead of the project level. The company no longer charged staff qualification to the project's budget, and now views it as an investment into quality and competitive advantage. As a result of this PMA, management realized the strategic importance of staff qualification and knowledge management—a truth that often gets buried in the hectic rush of Internet software business.

We received a lot of positive feedback from PMA participants in different companies. Particularly, they like that PMA offers a simple yet effective way to uncover both achievements and improvement opportunities. One developer at the satellite software company noted, "If you do a PMA on the project...you have to think through things," which is a crucial part of knowledge management. So, never leave a project without it!

References

1. C. Collison and G. Parcell, *Learning to Fly: Practical Lessons from One of the World's Leading Knowledge Companies*, Capstone, New York, 2001.
2. B. Collier, T. DeMarco, and P. Fearey, "A Defined Process For Project Post Mortem Review," *IEEE Software*, vol. 13, no. 4, July/Aug. 1996, pp. 65–72.
3. N.L. Kerth, *Project Retrospectives: A Handbook for Team Reviews*, Dorset House Publishing, New York, 2001.
4. A.J. Nolan, "Learning from Success," *IEEE Software*, vol. 16 no. 1, Jan./Feb. 1999, pp. 97–105.
5. T. Stålhane et al., "Post Mortem—An Assessment of Two Approaches," *Proc. European Software Process Improvement* (EuroSPI 01), ICSN, Bray, Ireland.
6. T. Dingsøyr, N.B. Moe, and Ø. Nytrø, "Augmenting Experience Reports with Lightweight Postmortem Reviews," *3rd Int'l Conf. Product Focused Software Process Improvement* (Profes 01), Lecture Notes in Computer Science, vol. 2188, Springer-Verlag, Berlin, pp. 167–181.
7. D. Straker, *A Toolbook for Quality Improvement and Problem Solving*, Prentice Hall International, London, 1995, pp. 89–98 and 117–124.

About the Authors

Andreas Birk is a consultant and software engineering professional at sd&m, software design and management. His special interests include software engineering methods, knowledge management, and software process improvement. He holds a Dr.-Ing. in software engineering and a Dipl-Inform. in computer science and economics from the University of Kaiserslautern, Germany. He is a member of the IEEE Computer Society, ACM, and German Computer Society. Contact him at sd&m, Industriestraße 5, D-70565 Stuttgart, Germany; andreas.birk@sdm.de.

Torgeir Dingsøyr is a research scientist at Sintef Telecom and Informatics research foundation in Trondheim, Norway. He wrote his doctoral thesis on "Knowledge Management in Medium-Sized Software Consulting Companies" at the Department of Computer and Information Science, Norwegian University of Science and Technology. Contact him at Sintef Telecom and Informatics, SP Andersens vei 15, NO-7465 Trondheim, Norway; torgeir.dingsoyr@sintef.no.

Tor Stålhane is a full professor of software engineering at the Norwegian University of Science and Technology. He has a MSc in electronics, and a PhD in applied statistics from Norwegian University of Science and Technology. He has worked on compiler development and maintenance and software reliability, and on software process improvement and systems safety. Contact him at Department of Computer and Information Science, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway; tor.stalhane@idi.ntnu.no.

Article:

Torgeir Dingsøyr, Nils Brede Moe, and Øystein Nytrø, “Augmenting Experience Reports with Lightweight Postmortem Reviews,” in *Third International Conference on Product Focused Software Process Improvement*, *Lecture Notes in Computer Science*, vol. 2188, F. Bomarius and S. Komi-Sirviö, Eds. Kaiserslautern, Germany: Springer Verlag, 2001, pp. 167 - 181.

Augmenting Experience Reports with Lightweight Postmortem Reviews

Torgeir Dingsøyr[1], Nils Brede Moe[2], and Øystein Nytrø[1,2]

[1]Department of Computer and Information Science,
Norwegian University of Science and Technology
Currently at: Fraunhofer Institute for Experimental Software Engineering,
Sauerwiesen 6, 67661 Kaiserslautern, Germany
dingsoyr@idi.ntnu.no

[2]SINTEF Telecom and Informatics,
7491 Trondheim, Norway
(Nils.B.Moe|Oystein.Nytro)@informatics.sintef.no

Abstract. Many small and medium-sized companies that develop software experience the same problems repeatedly, and have few systems in place to learn from their own mistakes as well as their own successes. Here, we propose a lightweight method to collect experience from completed software projects, and compare the results of this method to more widely applied experience reports. We find that the new method captures more information about core processes related to software development in contrast to experience reports that focus more on management processes.

1 Introduction

Many small and medium-sized companies that develop software seem to experience the same problems in several projects, like using more effort than originally planned, for example in the test phase. To reduce the impact of such problems, project managers would often like to know what preventive actions other projects in a similar situation has taken, and what the results of these actions has been. Other projects might experience technical problems with, say a compiler, that they know have appeared in the company before, but it is difficult to find which people were involved in solving the problem then. Very few companies have systems that will capture and share this type of information.

Another characteristic of small and medium-sized companies that develop software is that they are usually under strict time pressure. They do not have the time to invest in prevention of possible future problems in the project. Usually, projects are also pretty small, involving typically between 5-10 people, which also means that they cannot use a lot of resources on this.

Here we suggest a *lightweight* method to capture experiences from completed software projects, that is suitable for companies focusing on learning from their own experience. The reasons can be either to improve the quality of their products, be

F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2001, LNCS 2188, pp. 167-181, 2001.

more efficient, or to make the company a stimulating place to work. This is often referred to as *knowledge management* [1], and involves collecting, refining, packaging and distributing knowledge within a company.

Within software engineering, to focus on knowledge management has often been called to "set up an experience factory" [2]; an organizational framework for capturing and reusing experience from accomplished software projects. The idea here is to allocate resources for managing company-internal knowledge (the "experience factory"). –This unit should collect experience from ongoing and completed projects and make this available for others. It does not have to be a separate part of an organization, but some people should have a responsibility for it.

An interesting question in knowledge management in general, and particularly within software engineering, is how to collect, "harvest", or "make explicit" experience from projects so that they can be usable for others. What efficient methods exist, that do not require a lot of effort to make requisite knowledge available? In this paper, we are interested in looking at different lightweight approaches to capturing experience from projects (indicated with the bold arrow in Fig. 1), and in particular projects that are completed. This can be a way to make *tacit* knowledge [3] present in software development project *explicit*, and to store it in a knowledge repository or "experience base" to make it available as support for future projects. It can also be a source of data for improvement activities, which will be of interest to the management or "sponsoring organization". By "experience package engineering" in the figure, we mean to collect and make available experience on different topics in a way so that they are easy to use.

We will look at two different methods for capturing experience:

- *Writing Experience Reports*, reports written by project managers to sum up experience from the projects (single-author documents)
- *Conducting Postmortem Reviews*, a group process to investigate problems and successes in a project. We have developed a particular lightweight method, which does not require much effort.

To see differences between the methods, we have examined resulting reports from two software companies. Now, we first discuss the nature of experience, before saying what we mean by Experience Reports, Postmortem Reviews, and in particular *lightweight Postmortem Reviews.* We then give some context by describing the research project where this work was performed, as well as the companies where we collected the experience. Next, we limit the scope of this paper, and go on to present the research method applied here in section 2, results from each method in section 3, discuss them in section 4 and conclude in section 5.

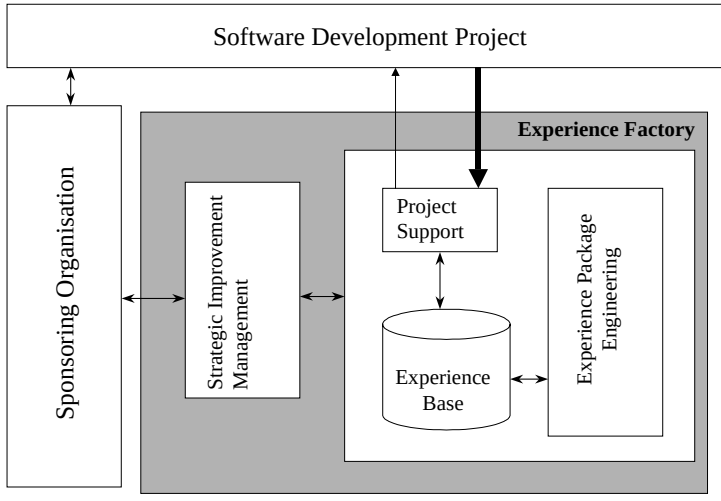


Fig. 1. Experience Factory organization, with experience capture from projects shown in bold. (Taken from the Perfect Project [4]).

1.1 What Is Experience, and What Forms Does It Take?

Let us start by defining a related, but more broad term, "knowledge". Here, we will use "knowledge" in a quite wide sense, meaning information that is "operational", that is, usable in some situation.

Experience in a strict sense is something that resides in humans, and that is not possible to transfer to others, because you have to experience it yourself to call it an experience. We will use the word in a less strict sense here, as "a description of an event that happened during project execution". An example is "Because of frequent changes of the requirements, it was hard to see what consequences they would have. This affected the testing of the software." Each such description, we will refer to as an "experience item".

A way to categorize experience or knowledge is to look at where it is applicable, and how easy it is to transfer. Novins and Armstrong [5] have suggested the following framework for categorizing knowledge: Experience that is collected can be used in a setting that we can divide into two categories: *local* and *global*. If experience is usable only locally, it is not applicable for many people or projects. If it is globally usable, many people can benefit from it. We can also use two categories of how transferable knowledge is. If it is easily transferable, we say that it is *programmable*. If it is difficult to transfer, we say that it is *unique.* Then we get the Tab. 1.

Yet another way to classify experience would be according to the topic they are about, for example to which part of the development process they are related, to which organizational role, or to what tools or special work methods. We developed one set of categories that are relevant to the projects we will describe later:

Table 1. Categorization of experience according to applicability and transferability.

	Local	Global
Programmable	*Easy to transfer, but suits only in some situations.*	*Easy to transfer, and usable in many situations.*
Unique	*Hard to transfer, and only relevant in some situations.*	*Hard to transfer, but relevant in many situations.*

- Processes: Contract negotiation, estimation, planning, specification, design, implementation, testing, administration and maintenance.
- Actors: Customer, Project Manager, Management, Developer
- Technology: (no subcategories).

We will come back to how we applied these categorization frameworks in the discussion in section 4.

1.2 Collecting Experience from Projects

Now, how are we supposed to collect experience from completed projects? We first give an overview of a frequently used method, then introduce Postmortem Reviews, and explain how the effort in conducting such can be reduced to make "lightweight Postmortem Reviews".

1.2.1 Experience Reports

A way to collect experience from a completed project is to write an "Experience Report". This document is usually written by the project manager after the project is finished. The report follows a fixed template, which makes it possible to compare reports from different projects. In one of the companies we worked with, the report is divided into two parts: The first part gives an overview of all the facts and numbers from the project: Start and finish date, size of contract, labour used, deviation from estimated work size, the number of source lines-of-code developed, documents produced, and the activities that contributed most to the excess consumption. The second part of the report describes problems during project execution with proposal for improvement. For each phase of the project there is a Problem Description and Proposal for Improvements. This information is represented as text. In the other company they only have part two with problem definitions and proposed improvements.

These reports are usually not longer than 10-15 pages in one company we worked with, and about 4 pages in the other. About 50% is devoted to each part.

1.2.2 Postmortem Reviews

There are several ways to perform Postmortem Reviews. Apple has used a method [6] which includes designing a project survey, collecting objective project information, conducting a debriefing meeting, a "project history day" and finally publishing the results. At Microsoft they also put quite much effort into writing "Postmortem reports", which are a bit more similar to what we have called "Experience Reports". These contain discussion on "what worked well in the last project, what did not work well, and what the group should do to improve in the next project" [7]. The size of the resulting documents are quite large, "groups generally take three to six months to put a postmortem document together. The documents have ranged from under 10 to more than 100 pages, and have tended to grow in length".

In a book about team software development, Watts Humphrey suggests a way to do postmortems to "learn what went right and wrong, and to see how to do the job better the next time" [8].

A problem with these approaches is that they are made for very large companies, who can spend a lot of resources on analysing completed projects. We work with medium-sized companies where 5-10 people usually participate in a project, ranging in size from about 8 to 50 manmonths. To suit this type of projects, we have developed a "lightweight" version of Postmortem Reviews.

1.2.3 Lightweight Postmortem Reviews

We have used Postmortem Reviews as a group process, where most of the work is done in one meeting lasting only half a day. We try to get as many as possible of the people who have been working in the project to participate, together with two researchers, one in charge of the Postmortem process, the other acting as a secretary. The goal of this meeting is to collect information from the participants, make them discuss the way the project was carried out, and also to analyse causes for why things worked out well or did not work out. A further description of what we did can be found in the "results" section.

Our "requirements" for this process is that it should not take much time for the project team to participate, and it should document the most important experience from the project, together with an analysis of this experience.

A description of another lightweight approach which seeks to elicit experience using interviews, and not a group process, is described by Schneider [9].

1.3 The Research Setting

Here we describe in what setting the research was carried out. We first introduce the research project we worked in, and then give a brief description of each of the companies and projects where we collected the empirical data for this paper.

1.3.1 The Research Project

The Process Improvement for IT industry (PROFIT) project is a Norwegian research project where researchers are working in tight cooperation with nine companies that develop software, with different process improvement initiatives. There are three main work packages: Process improvement under uncertainty and change, learning from experience, and process improvement through innovative technology and organization. The work which is reported here focuses on learning from experience. Some background information on earlier work on knowledge management systems within Norwegian software development companies has been published earlier [10].

1.3.2 Northern Software

Northern Software makes software and hardware for receiving stations for data from meteorological and Earth observation satellites. Since the company was founded in 1984, they have delivered turnkey ground station systems, consultancy, feasibility studies, system engineering, training, and support. Northern Software has been working with large development projects, both as a prime contractor and as a subcontractor. They possess a stable and highly skilled staff, many with masters degrees in Computer Science, Mathematics of Physics, and have what we can describe as a “engineering culture”. Approximately 60 people are working in the company, and the majority is working with software development. Projects are managed in accordance with quality routines fulfilling the European Space Agency PSS-05 standards, and are usually fixed price projects.

1.3.3 Southern Software

Southern Software is a software house specialising in advanced web-solutions, but not eBusiness. Examples are games, newspapers, customer services and resources, database access etc. Projects are usually small to medium sized. Larger projects are broken down in smaller packages by incrementally adding services to a web portal.

They have a heterogeneous staff; people with software as well as design background. Project teams often consist of many people with non-overlapping competence of webdesign- and programming, user interfaces, databases and transactions, software architecture, requirements and management. This means that communication costs are fairly high compared to overall project size.

The company recently started using Rational Unified Process (RUP) for some project parts (user requirements, management and testing). They recently assigned one full-time “knowledge manager” who receives all “knowledge harvest”-documents, user surveys and other project documents. This work has just started.

1.4 Scope

We could have looked at a lot of issues when comparing the two approaches discussed here. For example, we think that the Postmortem Reviews seemed to be

quite inspiring for the people who participated in the meetings, an effect you probably will not get with an Experience Report. However, we have limited the scope here to only discuss the results of the methods, that is, the written reports, and not the process of producing them, or any other effects they might have. What we will look for, is what kind of information we get out of each method.

2 Research Methods

The research performed is done in close collaboration with industry. We have participated in collecting experience from real software projects in a real environment, which means that we have little control of the surroundings. This is often referred to as action research [11, 12]. The benefit with this type of research is that the actual problems are very relevant to industry. A difficulty is that we have limited control over the surroundings, so the results might not be as generally applicable as when using for example experiments.

The material collected for this research is from two companies that we co-operated with in the PROFIT project. They were not selected arbitrarily; they were interested in working with the same topics that we were interested in. The projects we used as cases for lightweight Postmortem Reviews were selected by the companies. However, we asked them to select "typical" projects from their company, and also projects of a certain size.

The researchers have had two roles in this work: First, as a kind of process leader who have organized a meeting: Set the agenda in co-operation with industry and organized discussions. On the other hand, the researchers have been observers trying to document the process and the results of the meeting. In one company by using a tape recorder and transcribing important sections of the meeting, in another company by writing detailed minutes during the meeting.

When we analysed the material gathered, we had two reports, one experience report, and one post mortem review report from each company. An example part of an experience report is:

In this project the team members did not sit together, which complicated the communication between the members. In the last week of the project, however, the system track was placed together, which resulted in good communication and a stronger feeling of belonging to a team.

In our analysis, we would select sentences like the one underlined, and call them experience items. This was used in our later analysis as:

negative experience: physically separate

Another example is from a postmortem meeting, where one thing that was mentioned was:

Incremental development: Partial deliveries are motivating. Externally visible.

This was later documented in the report as:

The idea of incremental development, i.e. partial deliveries to the customer, worked very well. The customer became more aware of the project status, and was able to guide and enforce changes at an early stage.

Which was again summed up as an experience item:

positive experience: partial deliveries

Later, we would then do categorizations based on these experience items.

3 Results

Here we first present how we conducted lightweight Postmortem Reviews. Then we describe the contents of each of the reports from the Experience Report and the lightweight Postmortem Review, and give some examples of the experience that was collected. We only give examples from one company to save space, but will use results from both companies when we go on to discuss the differences between the results of the methods in section 4.

3.1 Lightweight Postmortem Review

We have used two techniques to carry out lightweight Postmortem Reviews. For a focused brainstorm on what happened in the project, we used a technique named after a Japanese ethnologist, Jiro Kawakita [13] – called "the KJ Method". For each of these sessions, we give the participants a set of post-it notes, and ask them to write one "issue" on each. We usually hand out between three and five notes per person, depending on the number of participants. After some minutes, we ask one of them to attach one note to a whiteboard and say why this issue was important. Then the next person would present a note and so on until all the notes are on the whiteboard. The notes are then grouped, and each group is given a new name.

We use a technique for Root Cause Analysis, called Ishikawa or fishbone-diagrams to analyse the causes of important issues. We draw an arrow on a whiteboard indicating the issue being discussed, and attach other arrows to this one like in a fishbone with issues the participants think cause the first issue. Sometimes, we also think about what was the underlying reasons for some of the main causes and attached those as well.

Now, for the Postmortem Analysis meeting, we organized it with the following sections:

1. Introduction: First, we (the researchers) introduce the agenda of the day and the purpose of the Postmortem Review.

2. KJ session 1: We handed out post-it notes and asked people to write down what went well in the project, heard presentations, grouped the issues on the whiteboard, and gave them priorities.
3. KJ session 2: We handed out post-it notes and asked people to write down problems that appeared in the project, heard presentations, grouped the issues on the whiteboard, and gave them priorities.
4. Root Cause Analysis: We drew fish-bone diagrams for the main issues, the things that went well and the things that were problematic.

We used a tape recorder during the presentations, and transcribed everything that was said. The researchers wrote a Postmortem report about the project, which contained an introduction, a short description of the project that we analysed, how the analysis was carried out, and the results of the analysis. The result was a prioritised list of problems and successes in the project. We used statements from the meeting to present what was said about the issues with highest priority, together with a fishbone diagram to show their causes. In an appendix, we included everything that was written down on post-it notes during the KJ session, and a transcription of the presentation of the issues that were used on the post-it notes. In total, this report was about 15 pages long.

The day after the meeting, we presented the report to the people involved in the project to gather feedback and do minor corrections.

A comparison of this method to another way of performing lightweight Postmortem Review is discussed in another paper [14], where you also find information on the resources required.

3.2 Results from Lightweight Postmortem Review

One result from the KJ session was two post-it notes grouped together and named "changing requirements". They are shown in the upper left corner of (some of) the results from the KJ process in Fig. 2.

Developer statements pertaining to this part of the KJ diagram:

"Another thing was changes of requirements during the project: from my point of view – who implemented things, it was difficult to decide: when are the requirements changed so much that things have to be made from scratch? Some wrong decisions were taken that reduced the quality of the software".

"Unclear customer requirements – which made us use a lot of time in discussions and meetings with the customer to get things right, which made us spend a lot of time because the customer did not do good enough work."

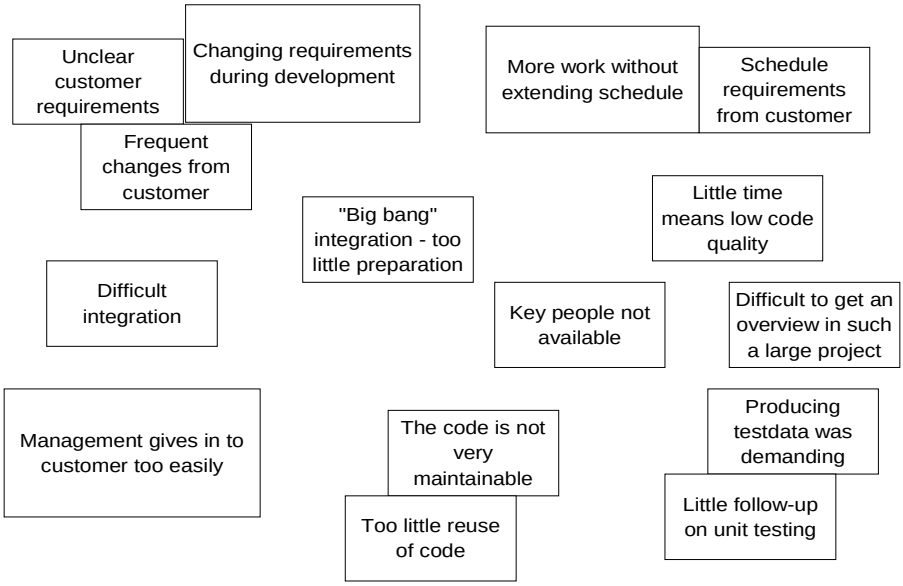


Fig. 2. Post-it notes showing some of the problems in a software development project.

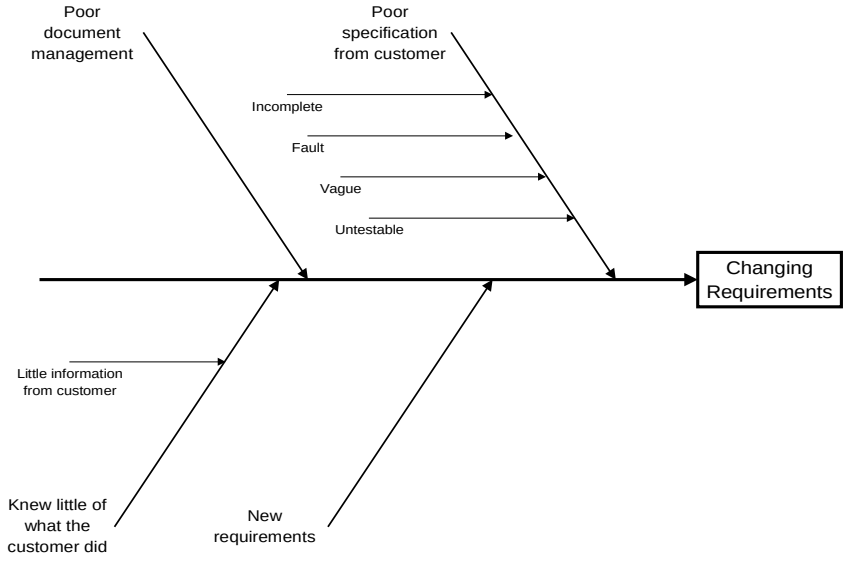


Fig. 3. Ishikawa diagram for “Changing Requirements”.

When we later brought this up again and tried to find some of the root causes for “changing requirements”, we ended up with the fishbone diagram in Fig. 3.

The root causes for the changing requirements, as the people participating in the analysis saw it, was that the requirements were poorly specified by the customer, there were "new requirements" during the project, and the company knew little of what the customer was doing. Another reason for this problem was that documents related to requirements were managed poorly within the company. In Fig. 3, we have also listed some subcauses.

In total, we found 21 experience items using this method at Northern Software, where nine were "positive", and 12 were "negative". At Southern Software, we found 18 "positive" and 8 "negative" experience items, making it a total of 26.

3.3 Experience Report

The project manager wrote the experience report. Of nine pages of information, five was introduction and description of the project, and four pages contained descriptions of "problems during project execution" with proposals for improvement.

An example of a problem is the "Architectural Design Phase", which was described in the following way: "This phase should have been carried out in ten weeks according to the plan. We carried out this according to the plan, and was just two weeks late (this delay was introduced earlier in the project). We still got changes from (customer department) in this phase. The negotiations about the contract did not start before this phase was finished, and we worked for a while without an architecture proposal or a contract in a period. Much of the management work was done through the architecture design phase: Contract negotiation, revising schedules, etc".

To solve the problems that appeared during this phase, the project manager has the following suggestions for improvement: "In total, this phase worked out ok. We were 300 hours behind schedule after this phase, which mainly came as a result of clarifying requirements from the customer. In the progress reports, I wrote that we should not give in to requests for changes in the estimates in the contract change notifications, or in the schedule. Experience indicated that we would get problems in the two next phases of the project, because the requirements were not stable, and the project was run according to (a standard process used in the company). I think we gave in to the customer in the negotiations about the schedule a bit easily, as well as the requirement on an intermediate delivery, which in a way worked as a template for the later negotiations. On the other hand, we got acceptance for the time estimates that we identified in contract change notification number 1 through 4."

At Northern Software, we found in total 28 experience items, where 27 were about problems, and just one about an activity that went well. The company produced this report with no inference from the researchers, who started working with them at a later stage. At Southern Software, we found 26 experience items, where 11 were "positive" and 15 "negative".

4 Discussion

Now, when we have seen a part of the reports from each of the two experience harvesting methods, can we say that there is any difference? Did the methods produce the same results? Could one method replace the other? To investigate this question, we studied the results in detail, and tried to group each of the experience items that were captured into one of the categories that we outlined in section 1.1.

We tried to apply the framework suggested by Novins and Armstrong, and to see if the experience could be said to be either *unique* or *programmable*, and *local* or *global*. Here, almost everything seemed to fit in the category *programmable* – the experience seemed to be pretty easy to transfer. To us, it looked like the experience would be valid for the whole company, that is, *global*. So with this framework, we could not distinguish between the results from the two methods.

We further found that almost all of the experience from both reports in both companies seemed to be of a kind that we can describe as "Problem understanding" [15, 16] – they were not as detailed as to use as a guideline, and we found little hard "facts" apart from in the introduction section in the Experience Report. Most of the information seemed to help "develop a better comprehension of a particular problem", and would be usable for "problem understanding".

Then, we tried to look at the topics that the experience was about. We made a list of *processes* in use in the companies, *actors* that would have different responsibilities, and also a category for *technology*, for experience related to tools, platforms and products. Using this type of categorization, and allowing one experience item to be related to both an actor, a processes or "technology", we found that the experience divided into the categories given in Fig. 4.

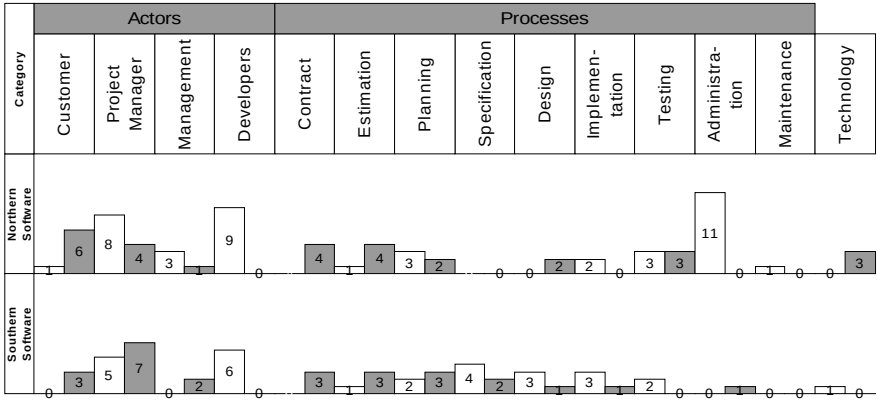


Fig. 4. Experience according to categories. Results from the Lightweight Postmortem Review in white and from the Experience Report in grey.

We see that most of the categories have experience from each of the methods, although the number varies. If we try to see which categories that have experience only from Postmortem Reviews, from both of the methods, and experience only

from the Experience Report, we find that the majority of experience categories are covered by both methods (6 categories). Postmortem Reviews covers one more (4) category than the Experience Reports (3) for Northern Software, but it is the other way around in the results from Southern Software. So it seems that the methods have covered slightly different issues.

To investigate this further, we examined the experience we had put in each category in more detail. Would these experience turn out to be the same, or different issues related to the same things in the same category? The results of this examination are given in Tab. 2. We have used the same set of categories as in Fig. 4, but replaced the name of each category with a number to save space. The number of categories that are "similar" means that "that number of the same experience items were found with both methods". The "total" figure refers to the sum of experience found with both methods.

From Tab. 2, we see that relatively few experience items are about the same issue. Only 5 experience items were found with both methods of a total of 69 experience items in Northern Software (some were put in several categories, so the sum of the "totals" in the table will be larger). At Southern Software, we found only three experience items with both methods, of a total of 49 experience items.

Table 2. Similarities of experience from each of the categories. "Total" is the total number of experience items in each category. Results from postmortem reviews are abbreviated PM, and experience report ER.

Category		*Actors*				*Processes*								
		1	2	3	4	5	6	7	8	9	10	11	12	13
Northern	PM	1	8	3	9	0	1	3	0	0	2	11	1	0
	ER	6	4	1	0	4	4	2	0	2	0	0	0	3
	Total	7	12	4	9	4	5	5	0	2	6	11	1	3
	Similar	*2*	*1*	*2*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
Southern	PM	0	5	0	6	0	1	2	4	3	2	0	0	1
	ER	3	7	2	0	3	3	3	2	1	0	1	0	0
	Total	3	12	2	6	3	4	5	6	4	2	1	0	1
	Similar	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*2*	*0*	*0*	*0*	*0*	*0*

So why is this? Why is it that we did not find a very large overlap in the results of the methods? One reason might simply be that by using a group process to elicit experience, we get to view what happened through several "different eyes". A reason in the Southern Software case might be that the content of the lightweight Postmortem Review was known when the Experience Report was written. But it was the other way around in the Northern Software case.

Another thing we found, independently of the categorization frameworks, is that most of the experience from the Experience Report at Northern Software were about problems, whilst we intended to get 50% about problems and 50% about successes in the Postmortem Reviews. A reason for this might be that the Experience Report is used more to explain what went wrong than the Postmortem Reviews, which had a

more precise goal of capturing experience that might be useful for others. At Southern Software, we did not find this in the Experience Report.

A difference we noted in Southern Software, was that the lightweight Postmortem Review would assign an experience to a situation and role, whereas the Experience Report would have a more managerial view of the experience, relating to the overall process. For example, enforcing formal change requests from the customer resulted in better cost control and planning from the management point of view. However, the developers considered that the main improvement was that they were relieved of direct, and frequent, customer interruptions.

5 Conclusions

Now we will sum up the conclusions we can draw from the discussion:

- The two methods find very little overlapping experience: We found more experience related to implementation, administration, developers and maintenance with the lightweight Postmortem Review. Whereas with the experience reports, we found more experience related to contract issues, design and technology.
- The experience items found with both methods seem to be usable for most projects within the companies, and seem to be quite easy to transfer.
- The two methods seem to find experience that can be used for problem understanding.

In all, it seems that both methods elicit information that are similar in style, but which is related to a bit different topics. If you are interested in issues related to the core processes of software development, lightweight Postmortem Reviews seems to be a better method than Experience Reports. If you are more interested in relations to the customer, Experience Reports seem to be a better choice.

Acknowledgements

We would like to acknowledge colleagues in the PROFIT project for providing a stimulating research environment, as well as our contact persons in Northern and Southern software. We are very grateful to Stefan Biffl at the University of Vienna, Reidar Conradi at the Norwegian University of Science and Technology as well as the anonymous reviews, for helpful comments on this paper. Furthermore, we are grateful to Geir Kjetil Hanssen at Sintef Informatics who managed part of the Lightweight Postmortem Review at Southern Software, and who made important contributions to the analysis. This work was supported by the Norwegian Research Council under project 137901/221.

References

[1] K. M. Wiig, *Knowledge Management Methods*: Schema Press, 1995.

[2] V. R. Basili, G. Caldiera, and H. D. Rombach, "The Experience Factory," in *Encyclopedia of Software Engineering*, vol. 1, J. J. Marciniak, Ed.: John Wiley, 1994, pp. 469-476.

[3] M. Polanyi, *Personal Knowledge - Towards a Post-Critical Philosophy*. London: Routledge and Kegan Paul, 1958.

[4] PERFECT Consortium, "PIA Experience Factory, The PEF Model,," ESPRIT Project 9090 D-BL-PEF-2-PERFECT9090, 1996.

[5] P. Novins and R. Armstrong, "Choosing your Spots for Knowledge Management," *Perspectives on Business Innovation*, vol. 1, pp. 45-54, 1998.

[6] B. Collier, T. DeMarco, and P. Fearey, "A Defined Process For Project Postmortem Review," *IEEE Software*, vol. 13, pp. 65-72, 1996.

[7] M. A. Cusomano and R. W. Selby, *Microsoft Secrets - How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*: The Free Press, 1995.

[8] W. S. Humphrey, "The Postmortem," in *Introduction to the Team Software Process, SEI Series in Software Engineering*. Reading, Massachusets: Addison Wesley Longman, 1999, pp. 185-196.

[9] K. Schneider, "LIDs: A Light-Weight Approach to Experience Elicitation and Reuse," presented at Second International Conference on Product Focused Software Process Improvement, PROFES 2000, Oulu, Finland, 2000.

[10] R. Conradi and T. Dingsøyr, "Software experience bases: a consolidated evaluation and status report," presented at Second International Conference on Product Focused Software Process Improvement, PROFES 2000, Oulu, Finland, 2000.

[11] D. J. Greenwood and M. Levin, *Introduction to Action Research*: Sage Publications, 1998.

[12] D. Avison, F. Lau, M. Myers, and P. A. Nielsen, "Action Research," *Communications of the ACM*, vol. 42, pp. 94-97, 1999.

[13] R. Scupin, "The KJ Method: A Technique for Analyzing Data Derived from Japanese ethnology," *Human Organization*, vol. 56, pp. 233-237, 1997.

[14] T. Stålhane, T. Dingsøyr, N. B. Moe, and G. K. Hanssen, "Post Mortem - An Assessment of Two Approaches," submitted to European Conference on Software Process Improvement, EuroSPI, Limerick, Ireland, 2001.

[15] C. W. Choo, *The Knowing Organization : How Organizations Use Information to Construct Meaning, Create Knowledge, and Make Decisions*: Oxford University Press, 1998.

[16] R. S. Taylor, "Information Use Environments," presented at Progress in Communication Science, 1991.