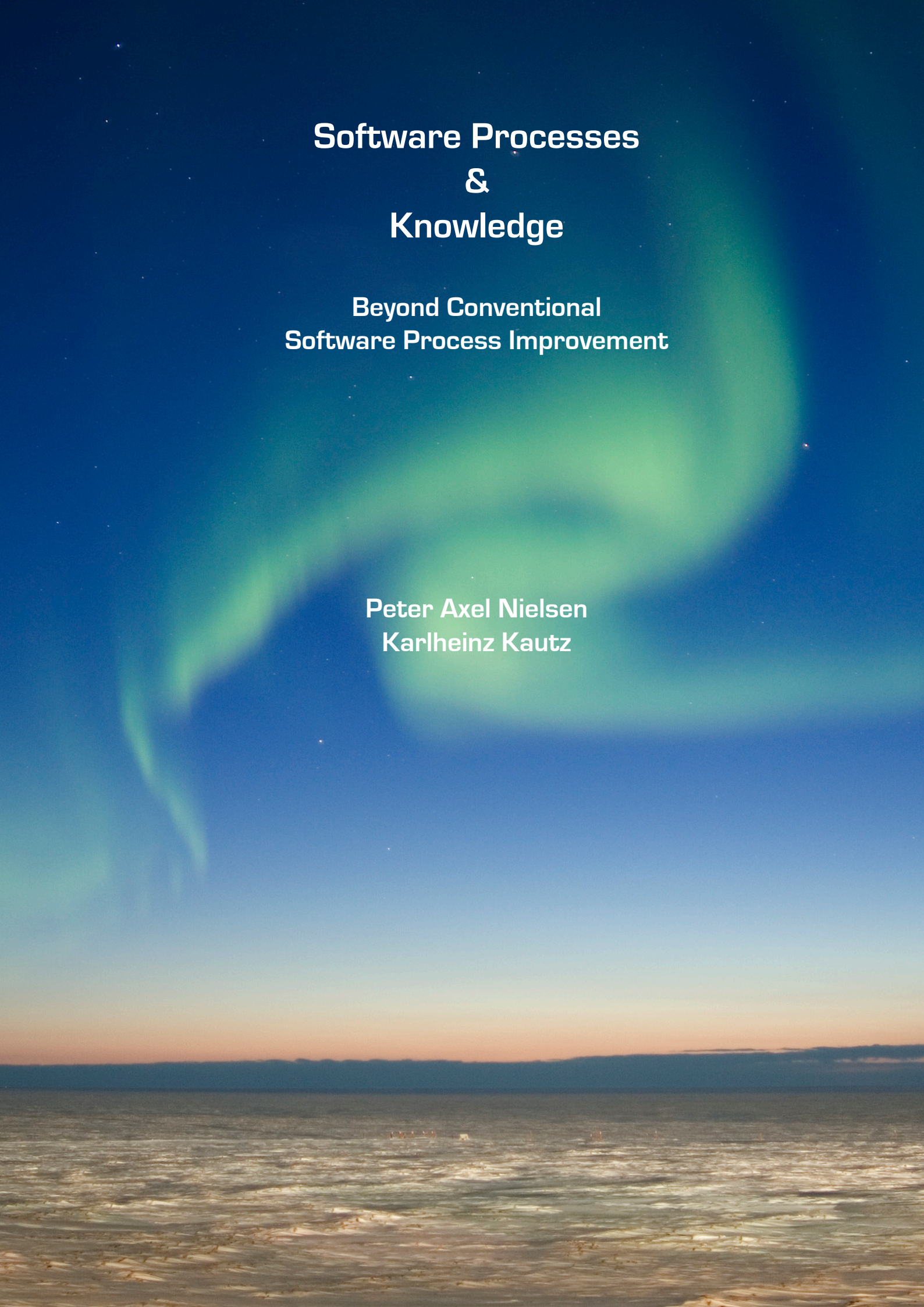

Software Processes & Knowledge

Beyond Conventional Software Process Improvement

Peter Axel Nielsen
Karlheinz Kautz

Software Processes & Knowledge

Software Processes & Knowledge

Beyond Conventional Software Process Improvement

Peter Axel Nielsen & Karlheinz Kautz

Software Innovation Publisher
Aalborg

Software Innovation Publisher
Aalborg University
Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg
Denmark

www.swi-publisher.dk

ISBN 978-87-992586-0-4

First printing, May 2008

Cover photograph: Twilight Aurora Borealis by Roman Krochuk / www.dreamstime.com

Table of Contents

Research Team

Software Process & Knowledge: Beyond Conventional Software Process Improvemen is based on a collaborative research effort. Following is a list of the contributors to this book, along with their affiliations during the project and their current e-mail addresses.

Thomas Elisberg	IT University Copenhagen	elisberg@itu.dk
Bo Hansen Hansen	Copenhagen Business School	bo@2hansen.dk
Jens Henrik Hosbond	Aalborg University	joenne@cs.aau.dk
Karlheinz Kautz	Copenhagen Business School	kautz@cbs.dk
Annemette Kjærgaard	Copenhagen Business School	amk.inf@cbs.dk
Lars Mathiassen	Georgia State University	lmathiassen@gsu.edu
Sune Dueholm Müller	Aarhus School of Business	sdm@asb.dk
Peter Axel Nielsen	Aalborg University	pan@cs.aau.dk
Jacob Nørbjerg	Copenhagen Business School	jacob@cbs.dk
Jan Pries-Heje	IT University Copenhagen	janph@ruc.dk
Gitte Tjørnehøj	Aalborg University	gtj@cs.aau.dk
Ivan Aaen	Aalborg University	ivan@cs.aau.dk

1

Software Processes and Knowledge

An Introduction

Peter Axel Nielsen and Karlheinz Kautz

This book is a result of the Software Process and Knowledge project which took place in Denmark in the period November 2002 until January 2006. In line with its predecessor, the Software Process Improvement project (January 1997 - December 1999, documented in Mathiassen et al., 2002), the project was a co-operation of three Danish academic institutions, Aalborg University, Copenhagen Business School and the IT University Copenhagen and three Danish IT companies. To improve the companies' software development processes and practices and to further the scientific understanding and building frameworks and theories on software development practice extensive, empirical research was performed in all three companies.

The book goes beyond the project as it also contains theoretical reflections independently of the three involved organizations and—as most of the researchers simultaneously also worked in similar projects with other organizations—we included results from co-operations with four other companies. Most companies wished to remain anonymous throughout the book. We respected this wish and named these companies with pseudonyms such as GlobeSoft, SpaceSoft, FinancialSoft, and SmallSoft, which still express some of their identity. For the largest co-operation partner, Systematic Software Engineering, it would not have made sense to use an alias since everyone interested in the IT business in Denmark know about it. More details about the companies can be found in the different chapters. Some companies appear in more than one chapter. Brief descriptions of

the companies in which the empirical work was carried out can be found in each chapter with the consequence that some companies are described more than once. We do not consider this a disadvantage—quite the opposite. As researchers who interpret what we find in the companies, these descriptions provide a broader view and give the reader overall a richer picture of the respective company and its context. Each chapter also includes a complete literature list for reference and further reading. This chapter design allows for the individual chapter to be read independently. In this introductory chapter we also like to use the opportunity to thank our co-operation partner. We hope that our joint endeavors have been as beneficial for them as they have been for us.

This book continues one important Scandinavian tradition of IS/IT research—while participating in what some might consider as mainstream research—we are also pursuing alternative paths to achieve results which are both relevant for theory and practice. Scandinavian researchers have raised their critical voices concerning conventional software process improvement regularly in the past. The publication *Improving Software Organizations—From Principles to Practice* (Mathiassen et al., 2002), which resulted from the predecessor project and had a clear practitioner-orientation, has already been mentioned above. A special issue of the *Scandinavian Journal of Information Systems,* Vol. 13, on trends in the research on software process improvement was directed towards the research community. This book combines both approaches, but has again a more academic orientation. However in the spirit of Scandinavian action research giving preference to relevance over rigor, it also provides practice-relevant results. Thus it should also be worthwhile for the interested practitioner.

Following this introduction the book consists of 9 chapters, each in the best academic tradition describing its underlying research method to allow for a validity test of its findings and results, which are organized in 3 parts. Part I discusses central frameworks of the book, the most prominent standard for software process improvement, the Capability Maturity Model (CMM), the concepts of standard, processes and practices and in extension of the notion of knowledge sharing presents a framework for identifying knowledge related improvement opportunities. More concrete knowledge-based techniques for software process improvement built the content of part II. In a crossover chapter it includes a framework and an approach to make sense of and negotiate different perspectives of practice and to develop joint improvement activities. Furthermore, it includes descriptions, discussions and practical applications of a technique for knowledge mapping, of the organizational structure of knowledge networks, of an approach to design software-based tools for knowledge sharing, and of a technique to analyze social networks as a starting point for improvement activities. The last part of the book contains two tales based on longitudinal research spanning time periods of 8 and 10 years, respectively. The first tale describes and studies the role of improvisation in the adoption of software process improvement approaches and techniques and the second contains the narrative account of one case company's journey from little maturity to the highest maturity level, level 5, in the CMM. This

chapter provides closure and also demonstrates the way the involved researcher group works as a collective. The book starts out with a conceptual chapter critically assessing and partly rejecting the assumptions behind the most prominent improvement standard, yet it finishes with a success story of an organization which has used the approach to its benefit.

In more detail the contents of the chapters is as follows: In chapter 2 Rose, Aaen and Nielsen provide a discussion about the widespread model for conventional software process improvement, the Capability Maturity Model (CMM). The authors analyze the key texts describing the CMM. They apply modern management theory and uncover eight interrelated management and organisational assumptions underlying the standard CMM. They put forward that these assumptions reflect management thinking in large production and manufacturing organisations particularly in America which might not be appropriate for other contexts and cultures. Thus, they identify the circumstances under which the model might provide the appropriate approach to software process improvement (SPI), but their critical assessment also identifies the conditions under which the CMM is most likely not to be appropriate. The analysis is based on an extensive literature review of existing SPI literature by Hansen, Rose and Tjørnehøj (2004); further considerations on this topic by members of the research team can be found in Ngwenyama and Nielsen (2003) and Aaen (2003).

In chapter 3 Müller, Nielsen and Nørbjerg argue that three important terms, namely standards, processes and practices are regularly confused in the literature. Consequently, understanding and implementing the different approaches which are discussed in the literature is difficult. They propose a simple framework based on a few distinctions: the distinction between software development practice on the one hand and defined and described software process on the other hand as well as the distinction between a company's defined processes and standards. The framework is applied in the analysis of three SPI projects, where it helps to resolve some of the conceptual mayhem. This shows that the framework might also support practitioners when planning and putting SPI standards and processes into practice. Aaen and Pries-Heje (2004) also discuss the challenges around standardizing software processes.

Chapter 4 deals with another term which has recently been related to SPI, namely knowledge management. Kautz and Kjærgaard focus in this chapter on knowledge sharing as a vital part of knowledge management. They introduce a theoretical framework which focuses on the relationship between knowledge, learning, communication and participation in action, and the role of social interaction and technical media in the knowledge sharing process. Drawing on an empirical study of knowledge sharing in a software development company, they discuss how knowledge sharing between the software developers unfolds and what hinders and supports knowledge sharing in software development. Although this chapter not explicitly relates to software process improvement the

identification of obstacles in the knowledge sharing process obviously provides opportunities for improvement.

In chapter 5 Kjærgaard, Kautz and Nielsen at the intersection of frameworks and techniques take the view that knowledge management is an ongoing sense making process in which realities are constructed and negotiated. They use this perspective to explore in one company how software developers and project managers make sense of the process of managing projects, and how this influences their perception of what should be included in a shared project management handbook as one sensible initiative to improve software development practice. The same authors have also applied the framework to understand the establishment of knowledge management as part of a more general scheme to improve practice in technology development firms (Kjærgaard and Kautz 2008).

A specific technique is presented in chapter 6. Kautz and Hansen describe a knowledge mapping technique to identifying and investigate how knowledge flows through organizations. They have assessed the technique's applicability in a pilot experiment in Systematic Software Engineering. The experiment tested and evaluated one possible way how the technique can be used to analyze an organization's knowledge sharing capabilities, as well as how it can point to opportunities for improvement and thus how it can support knowledge management based software process improvement. Pries-Heje (2004) has also developed a mapping technique; his technique focuses on the management of knowledge in IT projects while they are performed.

In chapter 7 Hosbond and Nielsen describes a case of software process improvement in which knowledge is shared through small knowledge tools. These tools are evaluated as prototypes and the consequences of their use are discussed.

Chapter 8 presents another technique to analyse communication and knowledge sharing in particular in small development organizations as a prerequisite for software process improvement. Software process improvement in such organisations is often problematic and standard-based approaches seem to overlook the particular problems of these organisations. In these companies the existing and non-existing social networks of the developers and managers often play a significant role. Nielsen and Tjørnehøj report from a case study where they have mapped social networks to understand a small organisation's internal, informal as well as formal communication and knowledge sharing. They show how social network mapping can be done and how the maps can be used to further software process improvement. Nielsen and Ngwenyama, (2002) and Nørbjerg and Ngwenyama (2005) provide additional insights on the role of social networks for software process improvement.

In the context of software process improvement in small software development organizations in Tjørnehøj and Mathiassen in chapter 9 tell a company's tale covering a ten year's period. With particular emphasis on knowledge creation, the authors investigate the role of improvisation, which is promoted as a means to overcome time pressure and lack of knowledge in rapidly changing environments. The company's experience suggests that

improvisation offers a great potential for innovation during SPI adoption in small software firms. The company's experiences suggest that managers can facilitate SPI adoption by sparking and leveraging improvisations through appropriate leadership. Kautz and Nielsen (2004) provide further experiences concerning the implementation of SPI innovations in organization.

In the last chapter of the book, chapter 10, Pries-Heje, Nørbjerg, Aaen and Elisberg report the story of Systematic Software Engineering covering the company's journey from CMM level 1 to the highest level of the standard, namely level 5. The authors use a simple organizational change model—unfreeze-move-freeze—as a theoretical lens to structure their tale. They provide the company's experiences over an 8 years period between 1997-2006 based on comprehensive documentation, meeting minutes, action research in the organization, as well as individual and group interviews. The tale contains describes the setting when the change process started, what were the promoters and barriers in changing the organization were, and what other companies can learn from this experience.

The book has been edited by two of the authors, Peter Axel Nielsen and Karlheinz Kautz, with the technical support from Annette Moss Nielsen and assisted by Johanna Vogel who provided help concerning the English language and proof reading. We would like to thank everyone involved in the completion of the book.

References

Aaen, I., (2003). Software Process Improvement: Blueprints versus Recipes. *IEEE Software*, 20(5): 86-93.

Aaen, I. & J. Pries-Heje, (2004). Standardising Software Processes: An Obstacle for Innovation? In: *Proceedings og IFIP WG 8.6 Conference*, Leixlip, Kildare, Ireland.

Hansen, B., J. Rose, & G. Tjørnehøj, (2004). Prescription, Description, Reflection: The Shape of the Software Process Improvement Field. *International Journal of Information Management*, 24(6): 457-472.

Kautz, K. & P. A. Nielsen, (2004). Understanding the Implementation of Software Process Improvement Innovations in Software Organisations. *Information Systems Journal*, 14: 3-22.

Kjærgaard, A. L. & K. Kautz, (2008). A Process Model of Establishing Knowledge Management: Insights from a Longitudinal Field Study. *OMEGA,* 36(2): 282-297.

Mathiassen, L., J. Pries-Heje, & O. Ngwenyama, eds. (2002). *Improving Software Organizations: From Principles to Practice.* Addison-Wesley, Reading.

Mathiassen, L. & P. Pourkomeylian, (2003). Managing knowledge in a software organization. *Journal of Knowledge Management*, 7(2): 63–80.

Ngwenyama, O. & P. A. Nielsen, (2003). Competing Values in Software Process Improvement: An Assumption Analysis of CMM from an Organizational Culture Perspective. *IEEE Transactions on Engineering Management*, 50(1): 100-112.

Nielsen, P. A. & O. Ngwenyama, (2002). Organizational Influence Processes in Software Process Improvement. In: *Proceedings of the European Conference on Information Systems*, S. Wrycza & K. Kautz, editors, Gdansk, Poland.

Nørbjerg, J. & O. Ngwenyama, (2005). Building and Maintaining Alliances in SPI: Implications for Organizating Effective SPI. In *Proceedings of the European Conference of Information Systems*, D. Bartmann, F. Rajola, J. Kalinikos, D. Avison, R. Winter, P. Ein-Dor, J. Becker, F. Bodendorf, & C. Weinhardt, editors. University of Regensburg, Regensburg, Germany.

Pries-Heje, J., (2004). Managing Knowledge in IT Projects. *The IFCAI Journal of Knowledge Management*, 4: 49-62.

Part I

Frameworks

2

Managerial and Organizational Assumptions in the CMMs

Jeremy Rose, Ivan Aaen and Peter Axel Nielsen

1 Introduction

Software development is both an important activity for developed countries' economic progress, and a complex, problematic and occasionally spectacularly, unsuccessful one. Software projects have a reputation for being late and over budget, and the resultant software often provokes complaints from purchasers and users. The organisations that commission and eventually pay for software naturally try to seek contractual guarantees that they will get a product with which they are happy; accordingly, software firms look for ways to reliably deliver outcomes that will satisfy their customers. However, increases in the complexity of the tasks facing software developers match various contributing developments in the fields that support software development, such as programming, mathematical and algorithmic support from computer scientists, technique and automation help from software engineers, and method support from information systems specialists. An important parallel development in the practice and literature of software development has been the emergence of support for the management of the software development task in the form of software process improvement (SPI). This literature has resulted in a variety of approaches for improving the organisation and management of software projects, including the BOOTSTRAP methodology (Kuvaja and Bicego 1994, p. 195; Kuvaja et al. 1994), TAPISTRY (a software process improvement approach tailored for small enterprises) (Kuvaja et al. 1999), SPICE (Software Process Improvement and Capability dEtermination), the Industrialized Software Organisation or Japanese Software

Factory Approach (Matsumoto 1981; Matsumoto 1987), the Application of Metrics in Industry (AMI) (Pulford et al. 1992), the Generic Software Factory or the Eureka Software Factory project (ESF) (Fernström 1991; Weber 1997) and the Experience Factory (Basili and Caldiera 1995; Basili et al. 1994).

Although many of these approaches share common characteristics, one SPI approach—the Capability Maturity Model (CMM) from the Software Engineering Institute (SEI) at Carnegie Mellon University—has tended to dominate both the debate and practice (Hansen et al. 2004). Many CMM improvement projects have been initiated in software organizations, resulting in so many reports of these projects, that the approach has achieved the status of the classic software improvement approach. Although CMM projects are numerous and widespread, it is difficult to find consistent and convincing objective research evidence that they are, in fact, successful in delivering better software; there are also a number of acknowledged problems reported with them.

We will argue that these problems stem from underlying assumptions, particularly management assumptions, in the approach. All practice techniques are dependent on sets of assumptions which are seldom articulated in the texts that describe them. These assumptions may concern the nature of software, software projects, organisation, management, learning, change, as well as underlying assumptions about science and technology and philosophical assumptions such as a particular ontology and epistemology. CMM, like other practice techniques, contains an interlinking series of assumptions, which we here (borrowing an appropriate technology metaphor) refer to as its assumption platform. Understanding CMM's assumption platform can help in the understanding of the strengths and weaknesses of the approach, and in determining for which circumstances it is appropriate.

In this chapter we investigate the platform of assumptions that underlie CMM and its more recent extension and replacement CMMI (Capability Maturity Model Integrated). We use the acronym CMM henceforth to cover both the original model and its developments and extensions. The investigation is based on the literature study reported in (Hansen et al. 2004). We analyze central texts describing CMM (mostly stemming from SEI) and derive eight interlocking assumptions characterizing these writings. We show that this assumption platform has roots (which the authors also acknowledge) in a particular tradition of management thinking. This tradition stems from Frederick Taylor's scientific management, and has other branches in method studies and the more recent business process re-engineering movement. The interlocking assumptions form a powerful web of ideas that are characteristic of a particular form of organization and management in the late industrial age. Although these ideas have a long history of success, they have primarily been associated with large manufacturing industries—that is, production in the factory. We show that most of the reported problems with CMM can be attributed to mis-matches between the assumption platform and the many different types of software organization that are characteristic of the modern software industry, few of which resem-

ble a late twentieth century industrial manufacturing plant. We question whether CMM's assumptions are universally appropriate, or are suited to changing characteristics in the software market such as globalization, the rise of the internet and rapid technology change.

2 Research Process

The research process contains two components:

1. Identification and textual analysis of key CMM sources.
2. Analysis of reported problems with, and critique of, CMM.

Key CMM sources include the documentation available at the web site of the Software Engineering Institute, and the relevant writings of Watts Humphrey (Humphrey 1989; Humphrey 1992a; Humphrey et al. 1999; Humphrey 1988; Humphrey 1992b; Humphrey 1995; Humphrey 1998; Humphrey et al. 1991) and his colleague researchers, particularly Paulk (Paulk 1995; Paulk 1996; Paulk et al. 1993a; Paulk et al. 1995; Paulk et al. 1993b). Textual analysis takes the form of grounded analysis of the sources and identification of implicit or explicit management assumptions common to the central sources. Assumptions are illustrated with typifying quotations.

The second part of the analysis involves articles relating to CMM which have been identified as descriptive or reflective in (Hansen et al. 2004). Many of the descriptive articles relate the case experiences of CMM implementations, including CMM situations experienced by practitioners as problematic. Reflective articles often involve a summary of experience or a theory-based critique of CMM. Although the general tone of this literature is positive (it is mainly written by SEI researchers and practitioners involved in CMM projects), various types of problems emerge as typical. Problem analysis summarizes the various kinds of difficulties and relates them to the underlying assumptions of CMM.

3 CMM: Assumption platform

In this section we set out to analyse the platform of assumptions which underlie the CMM approach, as documented in its core source texts.

CMM is formally defined as "a description of stages through which software organizations evolve as they define, implement, measure, control and improve their software process" (Paulk et al. 1995). The development of the model was based on Watts Humphrey's characterization of the software process (Humphrey 1988). The SEI worked with industry and government for four years to develop CMM before publishing it (Paulk et al. 1993a), and it is still evolving, partly in response to feedback from the practitioners us-

ing it. The CMM principle has also been extended to other areas (Konrad et al. 1996), including: the Software Acquisition CMM, System engineering CMM, Integrated Product Management CMM, and People CMM. In 1997 the proliferation of models led to an effort to integrate them into CMM Integrated (CMMI) (Ahern et al. 2001; Reifer 2002)

Humphrey's understanding of management acknowledges a debt to Taylor's account of scientific management, and its lineage can be traced through the time and motion and work method studies culminating in the 50's and 60's, the American quality movement of the 70's and 80's, and the business process re-engineering movement in the 80's and 90's.

"First-class people are essential, but they need the support of an orderly process to do first-class work," argues Humphrey (1989, p. ix) (quoting Taylor): "In the past the man was first; in the future the system must be first." Taylor's thinking or Taylorism separates thinking from doing in the work system, and allots the responsibility for thinking to managers (work system designers and controllers) and the responsibility for doing (executing the work system) to workers. Humphrey adopts two principles from Scientific Management:

> "The scientific selection, training and development of workers instead of allowing them to choose their own tasks and train themselves as best they could."
>
> "The development of a spirit of hearty cooperation between workers and management to ensure that work would be carried out in accordance with scientifically devised procedures." (Humphrey 1989)

4 Process Orientation

Process orientation is a natural consequence of Taylor's thinking in that work processes form the work system that must be scientifically established. In addressing classic problems in the development of software (hitting the deadline, holding the budget and delivering software quality), Humphrey focuses first on predictability though process:

> "We should first consider the characteristics of a truly effective software process. Fundamentally, it must be predictable. That is, cost estimates and schedule commitments must be met with reasonable consistency, and the resulting products should generally meet users' functional and quality expectations." (Humphrey 1989, p. 3)

If the organisational processes are right, argues Humphrey, the product will be timely, cost-effective, and meet user requirements. Although Humphrey's law of "mature processes and personal discipline enhance planning, increase productivity, and reduce errors," Endres and Rombach (2003, p. 195)) stress both of Taylor's original principles, and that the process orientation of CMMI has become the more dominant characteristic.

4.1 Hierarchical Management: Planning, monitoring, control

To the process orientation is added a generous dose of management control:

> "The outlines of a general solution are clear:
>
> 1. Apply systematic project management—the work must be estimated, planned, and managed.
> 2. Adhere to careful change control—changes must be controlled, including requirements, design, implementation, and tests.
> 3. Utilize independent software assurance—an independent technical means is required to assure that all essential project activities are properly performed." (Humphrey 1989, p. 64)
>
> "A 'managed process' is a performed process that is planned and executed in accordance with policy; employs skilled people having adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description." (CMMI 2002, pp. 42-43)

To Humphrey, SPI targets a generic process that can be modelled, controlled, measured, and improved. Improvements will only stick if reinforced by careful introduction combined with periodical monitoring:

> "The actions of even the best-intentioned professionals must be tracked, reviewed, and checked, or process deviations will occur. If there is no system to identify deviations and to reinforce the defined process, small digressions will accumulate and degrade it beyond recognition." (Humphrey 1989, p. 22)

4.2 Externally Imposed Generic Process Models

Although CMM is principally focused on achieving discipline in organisational process, it is not irrelevant which processes are adopted. An underlying (and highly traditional) software engineering model of software development was incorporated in the original CMM, and other process models underpin other parts of CMM. Thus, the process areas for the engineering part of CMM are: requirements development, requirements management, technical solutions, product integration, verification, and validation (CMMI 2002). Similar generic process models for other disciplines, such as project management and support, underpin other parts of CMM.

4.3 Documentation, Standardization and Institutionalization

In the CMM approach it is not enough for a software organization to have effective processes. Processes must be defined (documented) so that they can later be monitored and

improved. Nor is it enough that the processes are documented, they must be standard across the whole organization—that is, institutionalized—so that the work of the organization is routinely carried out according to the defined processes.

> "An organization's set of standard processes contains the definitions of the processes that guide all activities in an organization. These process descriptions cover the fundamental process elements (and their relationships to each other) that must be incorporated into the defined processes that are implemented in projects across the organization. A standard process enables consistent development and maintenance activities across the organization and is essential for long-term stability and improvement." (CMMI 2002, p. 42)
>
> "Not only must a policy be formulated, but it also must be documented and it must be used throughout the organization." (CMMI 2002, p. 36)
>
> "A managed process is institutionalized" (CMMI 2002, p. 43)

It is not enough that processes are institutionalized, but adherence must also be monitored and non-compliance addressed.

> "People not directly responsible for managing or performing the activities of the process typically evaluate adherence. In many cases, adherence is evaluated by people within the organization, but external to the process or project, or by people external to the organization. As a result, credible assurance of adherence can be provided even during times when the process is under stress (e.g., when the effort is behind schedule or over budget)" (CMMI 2002, p. 58)

The Ad Hoc (Immature) Organization	***The Mature Organization***
Software processes are generally improvised during the course of the project.	Software processes are defined and communicated to both existing and new employees.
Even if software processes have been specified, they are not rigorously followed.	Work activities are carried out according to the planned process.
The organization is reactionary and the managers are fire fighters.	Roles and responsibilities are clear throughout the project and across the organization.
Schedules and budgets are not based on realistic estimates and are routinely exceeded.	Schedules and budgets are based on historical performance and are realistic.
Product functionality and quality are often compromised to meet hard deadlines.	Expected results for cost, schedule, functionality and quality of the product are usually achieved.

Table 1: Characterization of immature and mature software organizations (Paulk et al. 1993a)

4.4 Organizational Progression to Maturity

In CMM, organizational development is characterized as a progression from immaturity to maturity. Immaturity is characterised as a state of near chaos where ad-hoc processes are improvised—the concept here has a clear negative connotation—by developers resulting in haphazard and inadequate outcomes, whereas maturity is characterised by process discipline (table 1).

Organizations achieve maturity by learning process discipline through management improvement initiatives, either through the traditional stages (levels 1-5):

> "To achieve lasting results from process improvement efforts, it is necessary to design an evolutionary path that increases an organization's software process maturity in stages. The software process maturity framework ... orders these stages so that improvements at each stage provide the foundation on which to build improvements undertaken at the next stage." (Paulk et al. 1993a, p. 22)

or continually: "As a software organization matures, costs decrease, development time becomes shorter, and productivity and quality increase" (Paulk et al. 1993a, p. 41).

4.5 Objective Measurement, External Verification and Certification

Software quality assurance involves two further principles which are developed further in CMM: the use of measurement in the pursuit of scientific objectivity in process control, and the establishment of an external monitoring and certification body based at SEI.

> "In a mature organization, managers monitor the quality of the software products and customer satisfaction. There is an objective, quantitative basis for judging product quality and analyzing problems with the product and process." (Paulk et al. 1993a, p. 19)

Measurement is thus incorporated into process thinking, and used for management's evaluation and target setting. Processes are brought 'under statistical control' (Humphrey et al. 1999). However, a further important use of measurement is in maturity assessment. Here an organisation's process maturity is measured against external standards to assess its maturity level, and to monitor its progress. External process maturity standards are determined by the SEI at Carnegie Mellon University, who thus verify software firms' level of maturity, providing certification. Certification acts as a measure of maturity progress and incentive for the organization, and as a badge of competence for software purchasers and commissioners (often large government organisations).

4.6 Goal-Directed Change through Rational Analysis and Learning

Institutionalization of the generic process in the building of software, plus measurement of outcomes and feedback to process designers, will facilitate: rational analysis of the organisations' processes, identification of sub-optimal process and process improvement. Learning is embedded in improved organisational processes. As better processes are documented and their implementation enforced, the organisations' practice improves:

> "At Level 5, new and improved ways of building the software are continually tried, in a controlled manner, to improve productivity and quality. Disciplined change is a way of life as inefficient or defect-prone activities are identified and replaced or revised." (Paulk et al. 1993a, p. 38)

4.7 Management-Sponsored Improvement Initiative

The software process improvement initiative is sponsored by management, and embodies the principles discussed above. Thus, the improvement initiative is also process oriented, controlled by management, documented and standardized. It is also defined by: 1) the level of process maturity, 2) the level to which it implements (but not explicitly) a traditional software engineering development model, and 3) the level in which it is measured and externally verified.

5 The CMMI Assumption Platform

The assumption platform for CMMI is summarized in table 2.

CMMI extends the application area of the original CMM model without changing the underlying assumptions. The strength of the assumption platform is the logical synergy of the components and their internal ability to reinforce each other—internal consistency. Thus, in the industrial model, a process gives management something to control, which then can be rationally analyzed with the help of measurements that form the basis for optimization which, in turn, gives maturity. In addition, the platform mirrors a set of powerful assumptions common in some types of production and service industries in the late industrial age. Process orientation is known to give effective quality control, for example, in companies that build for airlines. These assumptions can be attractive to software managers with an engineering background who see themselves as leaders (they envision and enact improvements) and also to those responsible for large investments in software (certification can be viewed as a process quality guarantee).

Assumption	*Description*
Process orientation	What is important about a software organisation is the way the work is organised (a process). Good (mature) software organisations have defined and repeatable processes which govern the way that they work (their capability) and lead to success.
Hierarchical management: planning, monitoring, control	Management has the responsibility for process standardization, learning and improvement. Software developers should execute the organization's processes according to the standardized models and descriptions.
Externally imposed generic process models	Software processes leading to effective software development are well understood and share generic features which can be externally documented. These generic process models are suitable for implementation in all software organisations.
Documentation, standardization and institutionalization	Good software organizations not only have standardized and documented processes, but those processes are institutionalized; that is, carried out throughout the organization. Software projects are therefore conducted in a similar fashion according to pre-defined process models.
Organizational progression to maturity	Software organisation improvement is understood as the movement from immaturity (undefined processes) to maturity (standardized and institutionalized processes) through a series of management led change initiatives.
Objective measurement, external verification and certification	The extent of process standardization and institutionalization can be measured, and the measurements used to achieve: (1) better standardization, (2) enforcement of processes, and (3) process learning leading to process improvement. Measurement represents objective knowledge about software processes, and maturity can be externally certified.
Goal-directed change through rational analysis and learning	Organizational learning is achieved by the rational analysis and optimization of processes which sets the goals for organizational development and change.
Management-sponsored improvement initiative	Software process improvement initiatives follow the principles outlined above: defined and documented externally imposed process, management-led, focused on maturity, objectively measured, analysis-oriented. Thus, the CMM model is an extensive, externally imposed process plan.

Table 2: The industrial model: CMM's assumption platform

The analysis of CMM's assumption platform in this chapter reflects the thinking of its creators: every implementation project is likely to adapt the framework to the needs and requirements of the individual situations in the light of that situation's particular cultural conditions and traditions. Thus, CMM projects are likely to be different, and also with respect to the extent to which they adopt the underlying assumptions. Organisations operating under entirely different management assumptions are unlikely to find the approach attractive, but may be forced to adopt it as a condition of entrance to particular markets (for instance the American defence software market). However, the assumptions are a self-reinforcing package, difficulty buying into one or other of the assumptions is likely to weaken the effectiveness of the total package. For instance, the rhetoric of management command and control is unpopular in Scandinavia, but there is little point in using a great deal of resources to specify and describe processes if their use is not later enforced.

Although CMM's industrial model assumption platform represents a powerful synergy, it tends to exclude many other kinds of management improvement approaches which are based on incompatible or contradictory assumptions. Within the school of approaches promoted by the SEI and the literature on software process improvement, there can be found many subsidiary ideas and approaches which are not obviously compatible with these central assumptions, and are correspondingly less developed, less adopted and less discussed. Examples are: personal discipline, people CMM, commitment, empowerment, improvisation, and community of practice.

The following section is concerned with problem analysis, especially in relation to the assumptions of industrial model software process improvement.

6 Problem Analysis: CMM and the industrial model

Our problem analysis is grounded in the existing CMM literature. At the heart of the perception that CMM is a qualified success is the observation that it has its own embedded culture. CMM (and its assumption platform) echoes the culture of large American software organizations (Paulish 1993; Sakamoto et al. 1996; Siakas and Georgiadou 2002). More specifically, it reflects the dominant ethos of large American production companies in the late industrial age. Evidence for this assertion can be found, for example, in the many documented problems with adapting classical SPI to smaller organisations (Brodman and Johnson 1994; Kelly and Culleton 1999; Larsen and Kautz 1997; Richardson 2002; Villalon et al. 2002; Ward et al. 2001). Furthermore, it is far from obvious that this industry model can easily be transferred to the development of software. Software development is not manufacturing: for instance, it rarely involves the mass production of identical copies of a small range of products (say an automobile). It could better be compared to new product development. Software developers are not production workers, but rather highly skilled (and extremely well-paid) knowledge workers. Software also differs from an

industrial product in that its cost is almost entirely focused in its development and maintenance; the cost of producing the second copy is negligible.

6.1 Process Orientation

The chief criticism of process orientation is not a critique of 'process' itself, but rather a critique of what is ignored when process becomes the dominant focus of an organisation. CMM reveres process but ignores people (Bach 1994): "a fool with a process is still a fool" (Messnarz and Tully 1999). Neither does CMM focus on underlying organizational problems that should be solved (Bach 1995). Furthermore, CMM effectively ignores quality in the software product (Trienekens et al. 2001; van Solingen et al. 1999), assuming an unproven generic link between quality in the process and quality in the resulting product. If the process is good, goes the assumption, then the eventual product will also be good. However, differing project and organizational circumstances may mean that a process that delivers a good product in one context may deliver a poor product in another context (van Solingen et al. 2000). CMM is also criticized for its lack of business orientation (Messnarz and Tully 1999), its poor awareness of organizational context (Nielsen and Nørbjerg 2001), and for ignoring technical and organizational infrastructures (Wang and Bryant 2002). Furthermore, SPI encourages an internal efficiency focus and thus market and competition blindness (Rifkin 2002). Managing a company by focusing on internal efficiency is like driving a car by looking at the instrument panel instead of the road ahead.

6.2 Hierarchical Management: Planning, monitoring, control

According to Jakobsen (1998), CMM is heavily associated with command and control hierarchal management and power structures. This assumes a top-down, centralized management style, and the traditional separation of thinking (i.e., managers) and doing (i.e., workers) (Aaen and Pries-Heje 2004). In organizational culture the CMM adheres to the hierarchical and rational types (Ngwenyama and Nielsen 2003). Control of the organisation rests in the hands of managers, where externally defined processes become proxies for control. In addition, CMM is distrustful of personal mastery, individualism and heroism (Bach 1994; Bach 1995). The organisation is envisaged as a machine bureaucracy (Mintzberg 1983). Hierarchical management has the disadvantage of discouraging personal initiative, commitment and individual responsibility since the principal responsibility of the worker is to follow the bureaucracy's rules (here expressed as processes). Individual initiatives (perhaps in regard to special circumstances) are regarded as non-compliance and punished. Bottom up improvement that is rooted in experience is (inadvertently) discouraged, as are non-stereotypical solutions to customized problems and appropriate (but non-compliant) reactions to changing external circumstances.

6.3 Externally Imposed Generic Process

Behind the external imposition of generic process models lies the assumption that, at some level, the development of software implies some basic processes which are universal to all software projects and firms, and that these can be described and optimized (the 'industry best practice' concept). These become a 'blueprint' (Aaen 2003) for the firm's own processes and improvement. The idea of 'canned' (one size fits all) processes is ridiculed by CMM critics (Bach 1994; Weinberg 1992-97). The chief characteristic of software projects is their diversity, and there is no silver bullet (Brooks 1987). Diverse project conditions require different processes (Deck 2001).

6.4 Documentation, Standardization and Institutionalization

According to Bach (1994), CMM reveres the institutionalization of process for its own sake. Standardization reduces the ability of the organisation to cope with change and diversity, whereas institutionalization of processes makes an organisation static. It takes a long time and much effort to achieve the enforcement of standard procedures and, accordingly, an equally long time to change them. The documentation is easily changed, but this is not likely to produce any change in institutionalized behaviour. This is, according to Bach, a fundamental misunderstanding of the nature of dynamism in organizations. Static processes fail to reflect environmental change (Ward et al. 2001), and certain types of software development (innovative and total solutions) may thus be impeded (Rifkin 2002).

Documentation and standardization of processes require a great deal of resources, however, not nearly to the extent that enforcing them institution wide would require, notwithstanding the subsequent learning about them and resulting changes. Most small software organisations sensibly choose not to bother. Some formalization and standardization helps the comparison of projects and quality control, but learning can often be achieved by more efficient means, such as the mutual engagement of developers across different projects, and knowledge and experience sharing. Lack of process formalization does not necessarily imply poor processes—the alternative assumption is that process understanding is shared through practice in a less formal manner, such as master/apprentice relationships, organizational story-telling, best practice accounts and practice communities. Process focus is here balanced with other software improvement concerns.

6.5 Organizational Progression to Maturity

The maturity concept is based on a popular concept of the growth and development of organisations as a life-cycle, where small organisations progress to be larger organizations and eventually come to dominate their industry sectors. The analogy with nature is somewhat dubious and lacking in empirical and theoretical justification. Maturity functions

best as an inspirational metaphor in CMM, something to aspire to, an incentive and a goal to be reached. Otherwise, it can only really be described as a state of organizational management (rooted in American late industrial norms) which Humphrey and his colleagues assume is desirable, in contrast to another managerial situation (immaturity) which they find much less acceptable. Maturity in this context can only really refer to process maturity, and the implicit value judgment (mature=good, immature=bad) is arbitrary.

Equally arbitrary is the assignment of particular process maturity characteristics to levels (stages of process maturity) (Bach 1994) which leads to so many problems in operation (Bollinger and McGowan 1991; Emam and Madhavji 1995) that an alternative (continuous) form of the CMM model is provided, thus removing the stage model design. A further problem with uncritical acceptance of the maturity metaphor is goal displacement: other valuable goals (software product quality, competence development, market position, technical excellence, improved salesmanship, innovation) are replaced by the more concrete, but arguably less valuable goal of achieving the next CMM level (Bach 1994). If maturity is a valuable organizational asset, then it should produce measurable organizational benefits, but there is little convincing evidence of this (Hansen et al. 2004), despite a concerted SEI effort to demonstrate it (Brodman and Johnson 1996).

6.6 Objective Measurement, External Verification and Certification

Measurement can be an effective way of learning, for example, in a large organisation with large numbers of independent software projects, but the assumption that measurement is by nature objective is obviously incorrect. All measurement programs have inherent difficulties: in assigning indicators to phenomena (particular intangible phenomena), in establishing the relative importance and weighting of measurements, and in interpreting the meaning of the resulting metrics. These problems are reflected in studies of the quantitative evaluation of process improvement (Henry et al. 1995). The addition of external certification adds a further complication because the association of numerical scores with process maturity factors (in order to determine, for example, what 'level' the organisation is at) adds both a further arbitrary element and a political dimension.

A CMM assessment may serve as an effective tool for the selling of consultancy services "you may think you know something about building software but look—you're level 1—what are you going to do about that?"; however, it is less clear whether it serves as an effective research vehicle for assessing process quality in software organisations, let alone wider aspects of the management of software firms. Organizational actors quickly learn how to report good metrics (Wohlwend and Rosenbaum 1994), and later how to manipulate them to their own political ends (Frederiksen and Rose 2003). Organisations learn how to satisfice (to look good at the tests without really doing much) in order to achieve the badge which certifies that they are competent. Software procurers use the certificate as

a mark of competence where a detailed knowledge of the supplier market would be a more reliable indicator.

Furthermore, metrics have an inherent steering effect in an organisation; employees concentrate on the activities that they are measured in, but to the detriment of other activities, and in more sophisticated managerial uses of metrics concentrate on this effect (e.g., Basili 1992), rather than metrics as a basis for rational analysis. In addition, measures of the CMM type are hard to map to organizational problems (Niessink and van Vliet 2001), and hard to align to organizational and marketing strategies (Rifkin 2001).

6.7 Goal-Directed Change through Rational Analysis and Learning

A rational analysis of historical data is mapping the past to predict the future. This is, of course, important but assumes stable conditions as well as project equivalence (one project looks much like another). Software development is characterized by rapid technological change and project diversity. Managing by these principles is equivalent to steering a car by looking into the rear view mirror. Difficulties in assuming that one project predicts the next are reported by Jørgensen and Sjøberg (2001). In addition, managers often do not act according to the results of rational analysis, but rather use it to justify decisions that they make on other (often political) grounds. CMM assumes rationality in change and ignores politics (Nielsen and Nørbjerg 2001). Moreover organizational learning (beyond the individual or group level) in SPI can be low or absent (Conradi and Dingsoyr 2000; van Solingen et al. 2000). Schneider (2002) investigates difficulties incorporating experience-based learning in SPI.

6.8 Management-Sponsored Improvement Initiative

CMM's blueprint for improvement resembles its underlying assumptions about how a software organisation should be managed—externally imposed, process oriented, management- and consultant-led, focused on maturity, objectively measured, and analysis-oriented. Jakobsen (1998) discusses the merits of bottom up process improvement, in contrast to CMM's top down approach. The scale and complexity of effective organizational change are often underestimated in software improvement projects (Sweeney and Bustard 1997). Externally imposed and management led initiatives do little for employee commitment, and have difficulty adjusting to internal organizational conditions and to on-going environmental change. They make little use of local learning and experience, and can create employee resistance where they are imposed on already pressurized workers. Casting an improvement program as an initiative can undermine continual improvement efforts made by self-directed software developers, by reinforcing management command and control through process management.

7 Discussion: The trouble with CMM

The best-documented instances of CMM success relate to large American suppliers of software for the defence industry (Hansen et al. 2004). One interesting research hypothesis to explore is that CMM is more successful where the assumption platform evident in the approach that matches the software organisation's underlying assumptions. CMM might be expected to be successful in organisations that are, or aspire to be: process oriented, hierarchical, based on traditional software engineering models, heavily standardized, large and established, measurement oriented, rational, goal directed and used to top-down change—that is, late industrial age production type organisations. CMM might be expected to be correspondingly less successful in organisations with other types of culture, tradition and aspirations. Software organisations with very different expectations may expect correspondingly more difficulty with the approach. If we summarize some of the organizational and managerial assumptions for modern software organisations that are not especially compatible with the industrial model, we could include:

- Organizational orientation: people orientation, organizational competence orientation, knowledge orientation, product orientation, contingency or problem-solving orientation, community orientation, technology orientation, innovation orientation.
- Management style: management as facilitation, inspiration, example, teaching, problem solving, culture development, self-organisation, virtual organisation.
- Underlying software development models: experience based models, theory based models, local problem solving, bricolage and improvisation.
- Communication style: informal knowledge sharing, community and culture reinforcement, reinforcement of informal communication, shared repertoire, shared myths and stories, competence transfer.
- Organisational strategy focus: adaptation of organization to market position, multidimensional (balanced) improvement, technological profiling, innovation leadership.
- Organisational evaluation style: description, story-telling, individual self-motivation, professional rivalry, subjective expert assessment, certification by consistent reputation, membership of a community.
- Organisational change style: technology-driven change, change management, organisational development, entrepreneurship, continuous change, bottom-up improvement.

Moreover, many of these alternative assumptions are reasonable responses to the conditions and trends of software development in developed countries, such as:

- Rapid technology development: along with consequent rapid re-skilling.
- Highly educated mobile workforces, as well as highly paid.
- Ubiquitous computing: wide variety of software and programming demands through all sectors of society, all industry sectors, many types of conventional and embedded software.
- Outsourcing of non-core competence work: including basic programming work.
- Globalisation: global software organisations serving many cultural contexts, leaving many niche software markets.
- Software organisation diversity: many types of software organisation serving many varied market demands.

The trouble with CMM could therefore be defined as its adoption of an historical industrial production management model that is poorly adapted to match the conditions or the diversity of modern software development.

8 Conclusions

In this paper we have analysed the textual sources on which CMM is based, and have specified the organisational and managerial assumptions which underpin it. The assumptions (process orientation, hierarchical management: planning, monitoring, control, externally imposed generic process models, documentation, standardization and institutionalization, organizational progression to maturity, objective measurement, external verification and certification, goal-directed change through rational analysis and learning, management-sponsored improvement initiative) form an interlocking platform for the improvement of the management of software firms. We have characterized this platform as typical of certain forms of large industrial production companies in the late industrial age, with their roots in the management theories of Frederick Taylor.

CMM can and has been adopted and used in a variety of cultural situations and organization types, but has also generated different kinds of problems and difficulties. We have summarized the problems reported in the literature, and have made the generalized observation that they are characteristic of applying an approach with a particular management philosophy in organisations that only partly share that philosophy. We have argued further that conditions and trends in modern software development necessitate a diversity of organisational forms and management styles. It therefore follows that CMM is far from a generally applicable improvement approach, and should instead be carefully targeted at certain types of software organisation which share, or at least can tolerate, its underlying philosophy. Moreover, CMM's historically focused management philosophy is likely to

make it increasingly inappropriate for developed societies in the information age; thus, further research into alternative kinds of help for improving more modern types of organisation is needed and long overdue.

References

Ahern, D. M., A. Clouse and R. Turner (2001). *CMMI Distilled: An Introduction to Multi-discipline Process Improvement*, Addison Wesley.

Bach, J. (1994). The Immaturity of the CMM. *American Programmer*, 7(9): 13-18.

Bach, J. (1995). Enough About Process: What we need are heroes. *IEEE Software*, 12(2): 96-98.

Basili, V. and G. Caldiera (1995). The Experience Factory Strategy and Practice. *Computer Science Technical Report*. College Park, MD 20742, USA, University of Maryland.

Basili, V., G. Caldiera and H. D. Rombach (1994). The Experience Factory. In: *Encyclopedia of Software Engineering - 2 Volume Set*, John Wiley & Sons, Inc.

Basili, V. R. (1992). Software Modeling and Measurement: The goal/question/metric paradigm. College Park, MD 20742, University of Maryland.

Bollinger, T. B. and C. McGowan (1991). A Critical Look at Software Capability Evaluations. *IEEE Software*, 8(4): 25-41.

Brodman, J. G. and D. L. Johnson (1994). What Small Businesses and Small Organizations say about the CMM. In: *16th International Conference on Software Engineering*. Sorrento, Italy.

Brodman, J. G. and D. L. Johnson (1996). Return on Investment (ROI) from Software Process Improvement as Measured by US Industry. *Crosstalk*, 9(4) .

Brooks, F. P. (1987). No Silver Bullet: Essence and accidents of software engineering. *Computer*, 20(4): 10-19.

CMMI, (2002). CMMI for Software Engineering, Version 1.1, Continuous Representation (CMMI-SW, V1.1, Continuous), Tech. report No. CMU/SEI-2002-TR-028, Pittsburgh, Software Engineering Institute.

Conradi, R. and T. Dingsoyr (2000). Software experience bases: A consolidated evaluation and status report. In: *Product Focused Software Process Improvement*, 1840, pp. 391-406.

Deck, M. (2001). Managing Process Diversity while Improving your Practices. *IEEE Software*, 18(3): 21-27.

Emam, K. E. and N. Madhavji (1995). The Reliability of Measuring Organizational Maturity. *Software Process - Improvement and Practice*, 1(1): 3-25.

Endres, A. and D. Rombach (2003). *Empirical Software and Systems Engineering: A handbook of observations, laws and theories*. Harlow, England, Pearson/Addison Wesley.

Fernström, C. (1991). The Eureka Software Factory: Concepts and accomplishments. In: *3rd European Software Engineering Conference*. A. Lamsweerde and A. Fugetta, eds, Springer-Verlag.

Frederiksen, H. D. and J. Rose (2003). The Social Construction of the Software Operation: Reinforcing effects in metrics programs. *Scandinavian Journal of Information Systems*, 15: 23-38.

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

Henry, J., A. Rossman and J. Snyder (1995). Quantitative-Evaluation of Software Process Improvement. *Journal of Systems and Software*, 28(2): 169-177.

Humphrey, W. (1989). *Managing the Software Process*. Reading, Massachusetts, Addison-Wesley Publishing Company.

Humphrey, W. (1992a). *Introduction to Software Process Improvemen*t. Pittsburgh, Software Engineering Institute.

Humphrey, W., W. A. Florac and A. D. Carleton (1999). *Measuring the Software Process: Statistical process control for software process improvement (SEI)*, Addison Wesley.

Humphrey, W. S. (1988). Characterizing the Software Process. *IEEE Software*, 5(2): 73-79.

Humphrey, W. S. (1992b). *Introduction to Software Process Improvement*. Pittsburgh, Software Engineering Institute. Technical Report, Pittsburgh, PA, Carnegie-Mellon University.

Humphrey, W. S. (1995). *A Discipline for Software Engineering*. Reading, Massachusetts, Addison-Wesley Publishing Company.

Humphrey, W. S. (1998). Three Dimensions of Process Improvement. Part I: Process Maturity: 1-7.

Humphrey, W. S., T. R. Snyder and R. R. Willis (1991). Software Process Improvement at Hughes Aircraft. *IEEE Software*, 8(4): 11-23.

Jakobsen, A. B. (1998). Bottom-up process improvement tricks. *IEEE Software*, 15(1): 64-68.

Jørgensen, M. and D. I. K. Sjøberg (2001). Software Process Improvement and Human Judgement Heuristics. *Scandinavian Journal of Information Systems*, 13: 99-122.

Kelly, D. P. and B. Culleton (1999). Process Improvement for Small Organizations. *Computer*, 32(10): 41-47.

Konrad, M., M. B. Chrissis, J. Ferguson, S. Garcia, B. Hefley, D. Kitson and M. Paulk (1996). Capability Maturity Modeling at the SEI. *Software Process - Improvement and Practice*, 2(1): 21-34.

Kuvaja, P. and A. Bicego (1994). BOOTSTRAP - a European Assessment Methodology. *Software Quality Journal*, 3(3): 117-127.

Kuvaja, P., J. Palo and A. Bicego (1999). TAPISTRY - A Software Process Improvement Approach Tailored for Small Enterprises. *Software Quality Journal*, 8(2): 149-156.

Kuvaja, P., J. Similä, L. Krzanik, A. Bicego, S. Saukkonen and G. Koch (1994). *Software Process Assessment & Improvement - The Bootstrap Approach*, Blackwell Publisher.

Larsen, E. Å. and K. Kautz (1997). Quality Assurance and Software Process Improvement in Norway. *Software Process - Improvement and Practice*, 3(2): 71-86.

Matsumoto, Y. (1981). *SWB System: A Software Factory*. Amsterdam, North-Holland.

Matsumoto, Y. (1987). *A Software Factory: An Overall Approach to Software Production*, IEEE.

Messnarz, R. and C. Tully, eds. (1999). *Better Software Practice for Business Benefit*. Los Alamitos, California, IEEE Computer Society.

Mintzberg, H. (1983). *Structure in Fives*. Englewood Cliffs, Prentice-Hall.

Ngwenyama, O. and P. A. Nielsen (2003). Competing Values in Software Process Improvement: An Assumption Analysis of CMM from an Organizational Culture Perspective. *IEEE Transactions on Engineering Management*, 50(1): 100-112.

Nielsen, P. A. and J. Nørbjerg (2001). Software Process Maturity and Organizational Politics. In: *Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective, Proceedings of IFIP WG 8.2 Conference*. B. Fitzgerald and N. Russo, eds., Boise, Idaho.

Niessink, F. and H. van Vliet (2001). Measurement program success factors revisited. *Information and Software Technology*, 43(10): 617-628.

Paulish, D. J. (1993). Case-Studies of Software Process Improvement Methods. Pittsburgh, Software Engineering Institute.

Paulk, M. C. (1995). The Evolution of the SEI's Capability Maturity Model for Software. *Software Process*, Pilot issue: 3-15.

Paulk, M. C. (1996). Effective CMM-Based Process Improvement. In: *6th International Conference on Software Quality*. Ottawa, Canada.

Paulk, M. C., B. Curtis, M. B. Chrissis and C. Weber (1993a). Capability Maturity Model for Software ver. 1.1. Pittsburgh, PA, Software Enginering Institute.

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

Paulk, M. C., C. V. Weber, S. M. Garcia, M. B. Chrissis and M. Bush (1993b). Key Practices for the Capability Maturity Model, Version 1.1. Pittsburgh, Pennsylvania, Software Engineering Institute.

Pulford, K., A. Kuntzmann-Combelles, S. Shirlaw and K. Harutunian (1992). *A Quantitative Approach to Software Management: The ami Handbook*. London, CSSE South Bank University.

Reifer, D. J. (2002). The CMMI: it's formidable. *The Journal of Systems and Software*, 50.: 97-98. Guest editor's corner.

Richardson, I. (2002). SPI Models: What Characteristics are Required for Small Software Development Companies? *Software Quality Journal*, 10(2): 101-114.

Rifkin, S. (2001). What Makes Measuring Software so Hard? *IEEE Software*, 18(3): 41-+.

Rifkin, S. (2002). Is process improvement irrelevant to produce new era software? In: *Software Quality - Ecsq 2002*. 2349: 13-16.

Sakamoto, K., K. Kishida and K. Nakakoji (1996). Cultural Adaptation of the CMM: A Case Study of a Software Engineering Process Group in a Japanes Manufacturing Factory. In: *Software Process*. A. Fuggetta and A. Wolf, eds, John Wiley and Sons, pp. 137-154.

Schneider, K. (2002). Experience based process improvement. In: *Software Quality - ECSQ 2002*. 2349, pp. 114-123.

Siakas, K. V. and E. Georgiadou (2002). Empirical measurement of the effects of cultural diversity on software quality management. *Software Quality Journal*, 10(2): 169-180.

van Solingen, R., E. Berghout, R. Kusters and J. Trienekens (2000). No Improvement Without Learning: Prerequisites for learning the relations between process and product quality in practice. In: *Product Focused Software Process Improvement*. Berlin, Springer-Verlag Berlin. 1840: 36-47.

van Solingen, R., R. J. Kusters, J. J. M. Trienekens and A. Van Uijtregt (1999). Product-Focused Software Process Improvement (P-SPI): Concepts and their application. *Quality and Reliability Engineering International*, 15(6): 475-483.

Sweeney, A. and D. W. Bustard (1997). Software process improvement: Making it happen in practice. *Software Quality Journal*, 6(4): 265-273.

Trienekens, J., R. Kusters and R. Van Solingen (2001). Product Focused Software Process Improvement: Concepts and experiences from industry. *Software Quality Journal*, 9(4): 269-281.

Villalon, J., G. C. Agustin, T. S. F. Gilabert, A. D. Seco, L. G. Sanchez and M. P. Cota (2002). Experiences in The Application of Software Process Improvement in SMES. *Software Quality Journal*, 10(3): 261-273.

Wang, Y. X. and A. Bryant (2002). Process-Based Software Engineering: Building the infrastructures - Editors' introduction. *Annals of Software Engineering*, 14(1-4): 9-37.

Ward, R. P., M. E. Fayad and M. Laitinen (2001). Software Process Improvement in the Small - A small software development company's most difficult challenge: Changing processes to match changing circumstances. *Communications of the Acm*, 44(4): 105-107.

Weber, H. (1997). *The Software Factory Challenge*. Amsterdam, IOS Press.

Weinberg, G. M. (1992-97). *Quality Software Management*. New York, Dorset House Publishing.

Wohlwend, H. and S. Rosenbaum (1994). Schlumberger's Software Improvement Program. *IEEE Transactions on Software Engineering*, 20(11): 833-839.

Aaen, I. (2003). Software Process Improvement: Blueprints versus recipes. *IEEE Software*(September/October): 86-93.

Aaen, I. and J. Pries-Heje (2004). Standardising Software Processes - An obstacle for innovation? In: *IFIP TC8/WG8.6 Seventh Working Conference on IT Innovation for Adaptability and Competitiveness*. Leixlip, Ireland, Kluwer Academic Publishers, Boston.

3

Standards, Processes and Practice

Sune Dueholm Müller, Peter Axel Nielsen and Jacob Nørbjerg

1 Introduction

Several well-known frameworks, such as the CMM (Paulk et al. 1993), the CMMI (CMMI-Product-Team 2002) or Bootstrap (Haase et al. 1994), as well as standards, such as the ISO-9000 family, the SPICE model, or the European Space Agency's standards, play crucial roles in software process improvement (SPI). These frameworks and standards all stipulate that a software producing organization should define, document, introduce, enforce, and continuously develop internal standards and processes for project management, software engineering, quality control and quality improvement to increase its maturity level. The concept of an organization's standard software process has been challenged since the ideas of SPI were first introduced. It has been argued that the development of quality software relies on good people, first and foremost (Bach 1995). It has further been argued that the CMM's perspective on processes relies on a replication of steps known from mass production (see chapter 2), but is unfit for the software design process where neither the end result nor the steps to achieve it can be known beforehand (Bollinger and McGowan 1991). A recent study found that process standardization in combination with systematic measurement and control may jeopardize a software organization's innovative capability (Pries-Heje and Aaen 2004).

On the other hand, proponents of capability maturity models, such as SW-CMM and CMMI, advocate that standard processes should not be seen as an end in itself, but rather as means. Professional software developers will need to interpret, modify, add to, and improvise around the standard processes to achieve the organizations' goals (Curtis 1998). Our view after reading the vast literature on SPI is that part of this disagreement

rests on concepts and is largely part due to a lack of clarity about these concepts and how they are applied.

This chapter recognizes this lack of clarity and agreement in the SPI literature on what a standard is, what a process is, what a model is, and what a standard process is. To rectify this, we propose a simple framework in which it is possible to model the relationship between external standards, internal processes, and practice.

Let us explain what we mean by a lack of clarity and agreement in the SPI literature. First, there is hardly any distinction made between process and practice. In their critique of CMM, Bollinger and McGowan (1991) seem to identify processes with practice, while Curtis (1998) addresses the difference between process and practice. We suggest that it is necessary to exercise a clear distinction between process and practice.

Second, major parts of the SPI literature make a distinction between a process and a standard process. It is a core idea in process models that a software organization should have a standard process which can be adapted to become a specific process (Humphrey 1990; Paulk et al. 1993; CMMI Product Team 2002). Both Bollinger and McGowan (1991) and Pries-Heje and Aaen (2004) argue against standardization which means, in effect, that they argue against a standard process. We suggest that both non-standard process and standard process are two possible properties of process.

Third, there is confusion about what a standard is and what a process is. In SPI a standard is often implicitly taken to be a standard process, but it then becomes difficult to distinguish (standard) process from the generally accepted models developed by standardization agencies like International Standard Organization (ISO) or by the Software Engineering Institute (SEI). These models are publicly available and are used as benchmarks against which to evaluate an organization's (standard) process. For example, the ISO's 9000-family of standards addresses quality management systems, while SEI's CMMI addtresses maturity of (standard) processes. We suggest that a standard is distinctly different from a process. We further suggest that a standard and a maturity model are two ways of formulating external requirements for internal processes.

This chapter is organized as follows. The following section explains how we define the three core concepts of standard, process, and practice and their relationships in a framework. Section 3 provides three cases where we apply the framework and in section 4 we use the framework to explore and highlight the different ways in which standards, processes, and practice are influenced by each other in the three cases. This section also demonstrates the usefulness of the framework in making sense of otherwise poorly understood relationships. The final section provides a summary and conclusions regarding the necessity of such a framework in SPI research and practice.

2 The Framework

It depends on the context what a standard is taken to be. An agricultural standard of organic products is obviously not the same as a software engineering standard. Consequently, different definitions of a standard abound. Various attempts at abstract and all-embracing definitions have been made. For example, the IEEE Standards Association's website defines 'standard':

> "A standard is a published document that sets out specifications and procedures designed to ensure that a material, product, method, or service meets its purpose and consistently performs to its intended use." (IEEE, 1998)

In the same vein, the International Organization for Standardization describes a standard as:

> "A document, established by consensus and approved by a recognized body, that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context"

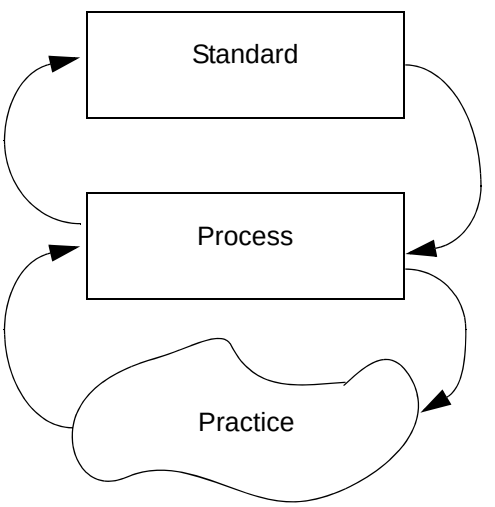

Figure 1: Conceptual model

Common to these two similar definitions is an emphasis on requirements to be imposed on either behavior or the outcome of production. The underlying rationale behind any standard is that by conforming to certain requirements, some desired outcome will ensue, e.g., homogeneity and uniformity of a process or a product. The benefits of such conformity include repeatability, predictability and quality. Consequently, standards can be

viewed as a vehicle for disseminating expert knowledge or even best practice in respect to attaining some desirable end (Jacobsson 2000). It also follows from the two definitions that standards are public, open, and external to any software producing organization.

On an even more abstract level standards "constitute rules about what those who adopt them should do, even if this only involves saying something or designating something in a particular way" (Brunsson and Jacobsson 2000, p. 4). Or simply put: "A standard is a guide for behavior and for judging behavior" (Abbott and Duncan 2001, p. 345).

In this chapter we limit ourselves to a discussion of a subset of standards, namely process standards (as opposed to product standards). Furthermore, the focus here will be on standards for software engineering and software development. A process standard provides directions for action and, as a minimum, specifies requirements that work practice must adhere to in order to be compliant with a particular standard (Brunsson and Jacobsson 2000, p. 4). Therefore, a standard can be anything from a set of process requirements to a process description detailing how to perform an activity. An example of the latter is the IEEE Standard for Software Reviews that can be used as a model for code inspections (IEEE 1998). An example of the former is the CMMI-SE/SW that spells out minimum requirements to software development and software engineering within different process areas and at different so-called maturity and capability levels (CMMI-Product-Team 2002).

Although defining the concept of standard is important, it is just as important to explicate and be aware of how standards relate not only to practice but also to software processes. Software processes are internal to the software producing organization, even though they may very well relate to the external standards. We are close to Humphrey when we understand a software process as "the set of tools, methods, and practices we use to produce a software product" (Humphrey 1990). We deviate slightly, however, when we argue that a process cannot contain a practice—process and practice remain distinctly different. What Humphrey means in his definitions is probably not practice as we are using it here; it is very likely that he means described experience and knowledge of how to perform a particular task. A software process can be described at several levels of detail, including a very detailed level. Irrespective of the level of detail, the concept of process in our context always means a process description.

A process is a description, which implies that a process is somehow explicated or documented. A process is distinctly different from practice. Practice is the actual unfolding of work over time within the organization as it is performed by individuals and by collectives of individuals. A shared practice or a common practice among a group of software professionals depends on a community-of-practice and should not be confused with a process.

In this chapter we illustrate how standards relate to software processes and how software processes relate to practice with figure 1. The external standards may influence the internal software processes which, in turn, may influence the practice. It is important to notice that there is no direct causal relationship, e.g., it would be wrong to expect that a

software process can be enacted leading to a particular practice. In fact, most of the criticism raised against maturity models (i.e., a standard) is that they assume processes to be mapped directly into practice. There is ample evidence that a process may or may not influence practice depending on many issues and aspects of software development (Bansler and Bødker 1993; Madsen et al. 2006; Kautz et al. 2007) and the influence may take many different shapes and forms.

As an example, a company choosing to utilize CMMI (a standard) may look at it and decide to describe several processes at level 2. The requirements for level 2 in CMMI stipulate several, but not all elements of the process when the company formulates it. The processes, in turn, may not initially lead to any changed practice, but if current practice is then compared to the descriptions in the processes there may be some areas where practice does not fulfill the expectations in the process. This may (or may not) gradually lead practitioners to change their practice, but most likely it will take more deliberate actions to make them change their practice.

It is also worth noting that current practice may influence a company's processes if, for example, it is decided that the practice is better than what is described in the processes. Further, the processes in several companies may be used to influence the external standards if the owner of the standard becomes convinced that the processes are better than what the standard requires.

The framework thus consists of the two distinctions combined with the influences that link the external standards to the internal processes and further influence the practice. The other way around, the practice may influence the internal processes, and internal processes or practices may influence the external standard.

3 The Three Cases

We apply the framework as a lens to analyze three cases. For each case we identify what standards, the processes, practices and the influences are.

3.1 Case One: GlobeSoft

GlobeSoft is a large Danish software development company offering mission critical solutions to a global market. The company spans many different industries and application areas, including space systems, radar systems, and defense and aerospace applications. The company comprises 5 business units and different staff functions, managing and supporting a broad project-based portfolio. Even though the business units have considerable latitude in selecting, planning and executing software development projects, all employees are expected to comply with the requirements of the GlobeSoft Management System (abbreviated GMS). The GMS consists of the company's software development process descriptions. The declared purpose of the GMS is to facilitate consistent execution of the estab-

lished management strategies and policies in a technically, as well as economically, optimal way.

In 2004 GlobeSoft initiated a both ambitious and company-wide process improvement effort. In part, due to market pressure and sales opportunities in the U.S., the CMMI framework was chosen as the maturity model for software process improvement. The process improvement project's long-term goal was to reach CMMI Level 3. The company chose to rely on internal assessments of process value and adherence.

The first goal was to reach CMMI Level 2. This first stage was planned as a two-stage project: (1) ensuring that the gap between existing internal process descriptions in the GMS and the CMMI was determined and discrepancies were reconciled; (2) executing an organization-wide implementation strategy based on the GMS. Seven process area teams (PAT), corresponding to the seven process areas at CMMI Level 2, were given the task of updating and supplementing core company policies and procedures as needed. During the implementation the newly improved processes in the GMS were deployed in 2 steps. First, processes were used in selected projects from different business units yielding valuable feedback about the further need for process editing. Second, when apparent obstacles had been removed and when all critical issues had been addressed, the new processes were implemented widely in the company.

GlobeSoft's management considered reliable measurement to be a key ingredient in their effective management of the process improvement project. Therefore, different measurement techniques and tools were used to ascertain compliance with the GMS's processes (i.e., the newly defined or updated standard processes).

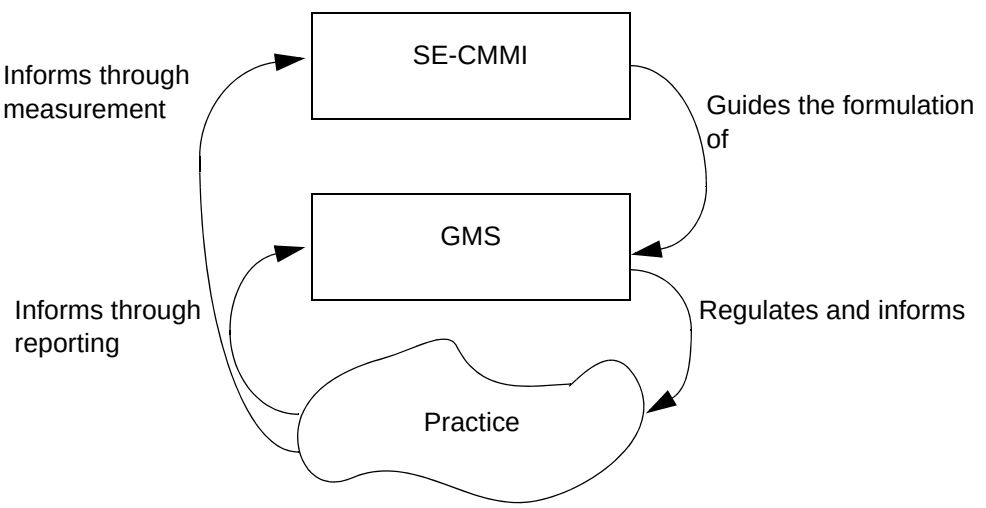

Figure 2: GlobeSoft case

Figure 2 shows the three conceptual levels of the standards, processes, and practices and the types of influences in GlobeSoft's SPI project. The SE-CMMI model has been chosen as the external standard against which all internal processes and practices are to be evaluated. The GMS contains the company-wide software processes. The software development practice unfolds at the level of projects. Interpreting and adapting the model requirements of CMMI Level 2 compliant processes to the organizational constraints and business needs of GlobeSoft have shaped the GMS. These processes were implemented as GMS policies, procedures or instructions depending upon their application domain and level of detail. The influence from the external CMMI standard on the internal processes in the GMS is illustrated in figure 2 by the 'guides the formulation of' arrow. At the project level, project-specific processes are instances of the processes in the GMS, adapted to the unique circumstances of the specific project. These project-specific processes then inform the actual practice in the project. The influence from the GMS to practice is illustrated by the 'regulates and informs' arrow.

Because the influence between: (1) CMMI and the GMS and (2) between the GMS and practice is open to many interpretations, it was decided in GlobeSoft to supplement the open interpretations with systematic measurements. A measurement strategy that met management's information needs was implemented. The purpose of the measurement strategy was to evaluate compliance between processes and practice. The compliance between internal processes in the GMS and the external standard CMMI was assessed by the PATs assisted by external consultants. The compliance between the GMS and the projects' practices was assessed by audits where systematic evaluations of work products were performed.

Using a questionnaire tool called Electronic-Based Assessment Tool (EBAT) to measure a project's degree of compliance with CMMI Level 2, GlobeSoft assessed the compliance between the external CMMI model and the projects' practices. Thus, the different types of measurements addressed the various risks associated with interpretation of standards and processes. This measurement strategy explains the remaining two influence arrows in figure 2. First, the work product audits providing feedback to the GMS when inoperable processes were identified (illustrated by the 'informs through reporting' arrow). Second, the EBAT assessments have an educational purpose to influence practice by creating awareness among project managers about best practice as suggested by the CMMI.

3.2 Case Two: SpaceSoft

SpaceSoft is a Danish company that produces software and systems solutions for the European Space Agency (ESA). The products include on-board software and hardware as well as simulation and testing environments for systems and software from other contrac-

tors. The company does not contract directly with ESA but works as a sub-contractor for other (mostly larger) European companies in the space industry.

SpaceSoft's SPI project began in early 2003 and was staffed (part time) with a group of project managers. Researchers had regular meetings with this group, discussing project progress, giving presentations, and conducting workshops and interviews. The CEO of the company was also present at the meetings with the researchers.

SpaceSoft's market puts high demands on the quality of the delivered products, and as a subcontractor to ESA, the company's software processes and products must comply with the standards described in various ESA documents. It was, therefore, more important for SpaceSoft's SPI project to ensure continued compliance with ESA standards, than to reach a specific maturity level as described in, e.g., CMMI or Bootstrap. Hence, a complete maturity assessment of the company's software processes was not performed. Instead, a series of meetings and workshops during the first half of 2003 were dedicated to assess the compliance between software processes and practices and ESA's standards, and from that assessment to prioritize improvement areas. The assessment showed that SpaceSoft more or less referred to the ESA standards as their process framework, and that SpaceSoft had very few locally described processes. Most project managers and software developers had, however, too limited knowledge of the content of the ESA standards and software engineering; further, project management practices varied considerably across projects. This resulted in rework, delays, and little exchange of ideas and good practices between projects.

With that background, the SPI project decided to produce handbooks (see chapter 5) for various development project roles, e.g., project managers, software engineers, and quality managers. The first version of each handbook should describe responsibilities and activities for a particular role as well as possible processes, tools, and techniques. The process descriptions should be sufficiently rigorous or detailed to comply with ESA standards, but not more.

A training program for each role in combination with on-going self-assessments and experience sharing among developers and project managers was designed to stimulate continuous improvement, reduce variations in practices across projects, as well as contribute to the on-going development of the handbooks. The initiative was furthermore expected to improve the software professionals' knowledge of the ESA standards and their impact on SpaceSoft's software processes.

The SPI project first addressed the project management role, and the production of a project manager handbook commenced. The first version of the handbook: (1) described the responsibilities of a project manager, activities needed to fulfill the responsibilities, the purpose, input and output of activities, and (2) provided links to further material in the ESA standards in a common repository. For each activity, the handbook also listed relevant ESA requirements and local (SpaceSoft) interpretations of these requirements. The section on project planning recommended, for example, that project managers do not pro-

duce functional breakdowns of projects as specified in the ESA standards, but rather document functionality through use case diagrams (PM Handbook, draft 1. September 2004, p. 3; further elaborated in v. 3; October 2004). Similarly, the handbook recommended use of spreadsheets for work breakdown structures and estimation instead of the tool recommended by ESA.

Interestingly, two perceptions of the purpose of the handbook emerged. On the one hand, project managers and developers in the SPI project group intended to use the handbook to create an improvement cycle, where the handbook would increase project managers' awareness of their responsibilities and stimulate ongoing dialogue and debate towards changing practice as well as the handbook itself. The CEO, on the other hand, saw the handbook as a guide for practice; i.e., a detailed process to be followed as closely as possible in practice.

Thus, the following conceptual elements and influences are found in SpaceSoft (see figure 3). The ESA standard regulates the process description in the handbook. The handbook then informs or guides practice, depending on the perception of the handbook's purpose. Finally, experience from practice feeds back into the formulation of the internal processes in the handbook.

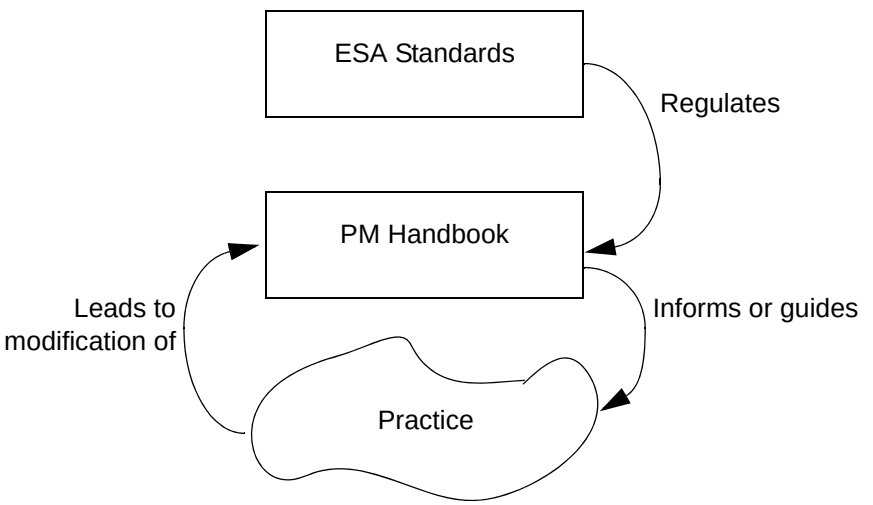

Figure 3: SpaceSoft case

3.3 Case Three: FinancialSoft

FinancialSoft is the IT department of a large bank. At the time of our analysis FinancialSoft employed more than 2000 developers and project managers in several locations. Fi-

nancialSoft had more than 150 project managers, some very experienced, but most had little experience in this management role. A number of problems had occurred in recent time that made it apparent that project management had to be improved. Examples include: it was difficult to find experienced project managers for risky and critical development projects, there were project overruns of more than 200% for even very small projects, and many project managers were unaware of best practice in project management. To remedy this state of affairs, two improvement efforts were launched in parallel: a project manager education and the FinancialSoft Project Management (PM) processes, in FinancialSoft's own jargon misleadingly called Standard. The focus here is on the FinancialSoft's PM processes.

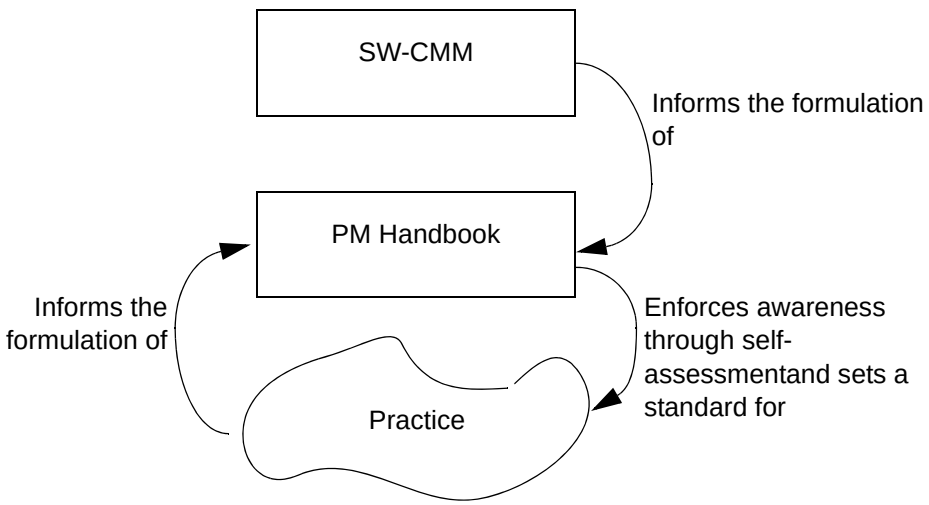

Figure 4: FinancialSoft case

Some argued that if all project managers had a 'common' practice, then all problems would be solved. Others argued that this would be an ideal situation never to be achieved, and they were altogether quite pessimistic about improving project management practices. The chosen approach came from the idea that it does not matter in what manner project managers practice, as long as they meet certain 'standards,' in the sense of levels of qulaity. Thus, rather than adhering to a norm, the idea in FinancialSoft was to think of a 'standard' as a bar to pass, i.e., to be above the bar, or set a benchmark against which to measure specific project management practices. Although it was called the FinancialSoft PM Standard, it was in effect a set of 10 processes described in an overall manner with a strong focus on process goals; a proper name would have been the FinancialSoft PM Process.

The SW-CMM 1.1, Level 2 served as the external standard, but it was argued that FinancialSoft had additional needs and maybe even better processes than SW-CMM. It was thus decided to develop FinancialSoft's own processes for good project management. During a 2-day seminar with very experienced project managers the draft of the FinancialSoft PM Processes was formulated. The first day was spent on going through the key process areas in SW-CMM Level 2 in detail and picking up the necessary and useful concepts, best practices, and values. In this way inspiration was sought in SW-CMM. The other major influence on the FinancialSoft PM Processes came directly from practice through the experienced project managers. The FinancialSoft PM Processes in the end had the same six areas as level 2 of the SW-CMM but, in addition, there were four other areas, e.g., covering the project manager's responsibility for business development in the bank. Each of the ten areas was described in one page. The way in which these ten processes were described did not differ from other internal processes in other software organizations.

The FinancialSoft PM Processes influenced practice directly in two ways (see figure 4). First, management talked about it at seminars, and all project managers were told to read it, understand it, and practice it in ways that would ensure passing the bar. Second, a tool for self-assessing project management practices was developed. The tool would, in less than a hundred questions, show a project manager in which areas his or her practice did not live up to the described process.

4 Analysis and Discussion

The three cases show very different uses of standards and processes. In particular the three cases show different influences taking place between the framework's three conceptual elements. Table 1 summarises the framework's elements and influences for the three cases.
The three cases map onto the framework in different ways. That means not only that the standards in the three cases are different, but they have also been chosen for different reasons. For GlobeSoft, the primary purpose was to become certified at CMMI Level 3, hence CMMI was chosen as the external standard. For SpaceSoft, the primary purpose was to demonstrate and document compliance with ESA's standards. For FinancialSoft, the primary purpose was to learn from the knowledge embedded in SW-CMM in the development of an internal benchmark. Hence, in all cases the purposes were varied and different standards were chosen; accordingly, what was then chosen as processes and what was described varied in the three cases. However, in all three cases the simplicity of the framework helped us in making sense of what was external, what was taken to be a process and which practice it related to.

	GlobeSoft Case	*SpaceSoft Case*	*FinancialSoft Case*
Standard	SE-CMMI	ESA standards	SW-CMM
Processes	GlobeSoft Management System (GMS)	PM Handbook	FinancialSoft PM Process
Practice	Software development in projects	Project management	Project management
Standard → processes	Guides the formulation of processes	Guides the formulation of processes	Informs the formulation of processes
Standard → practice	Informs through measurement	-	-
Processes → practice	(1) Regulates practice (2) Informs practice	(1) Informs (2) Guides in detail	(1) Sets a benchmark (2) Increases awareness through self-assessment
Practice → processes	Informs the formulation of processes through reporting	Leads to modification of processes	Informs process formulation based on experience

Table 1: A comparison of the standards, processes, practices and influences

Not only was the meaning ascribed to 'standard' and 'process' different in the three cases, the influences between the three levels of the framework were also different. This is illustrates the contribution of the framework in several ways. First, the three levels in the framework allow us to identify and attribute meaning to the influences. If we did not distinguish between process and practice, we would not be able to see the influences from process to practice, and back again. Second, the infls from the external standard to the internal processes depend largely on the primary purpose for bringing in the external standard. The difference between: (1) 'guides' in the GlobeSoft and SpaceSoft cases and (2) 'informs' in the FinancialSoft case demonstrates this difference. Third, the character of the influences from internal processes to practice shows how managers and developers in each of the cases view the difference between process and practice. In SpaceSoft some saw processes as step-wise procedures for practice, while others saw the processes as just a very abstract description that would never capture the complexities of the managers' and developers' competence. Fourth, the character of the influences from practice to internal processes is important to identify in order to assess whether there is a proper feedback mechanism in place. The feedback mechanisms in the three cases vary from a very rigorous ap-

proach in GlobeSoft to a very loose and informal one in SpaceSoft. Fifth, whether there is an influence directly from the external standard to the practice (as in the GlobeSoft case) shows whether there is a compliant and sophisticated measurement mechanism in place. For SpaceSoft, it is not as important that practice strictly adheres to the ESA standards, as long as the processes are compliant; the allowed variety between the ESA standards and the current practices is simply greater.

5 Conclusion

The framework introduced in this chapter has contributed to a richer understanding of (1) processes, standards and practice on a conceptual level and (2) the contextual relationship between these concepts. Through the analysis of three cases we show how processes and standards influence practice in different ways, depending on how they are interpreted and used by actors situated in varying organizational contexts.

The framework and the analysis contribute to research and value and impact of standards in software development and SPI by demonstrating that what is taken to be a standard and processes is situational, and that therefore all utilizations of standards and processes (and their implications) should be analyzed in a concrete context. We therefore suggest that further research into the value and implications of standards and processes in SPI must take the specific interpretations of, and relationships between, standards, processes, and practices into account.

Using the framework in the study of SPI projects can furthermore lead to a better understanding of alternative approaches to process description, as well as how processes and practice can influence each other, and thus, helps practitioners choose between alternative strategies in SPI.

References

Abbott, K. W. and S. Duncan (2001). International 'Standards' and International Governance. *Journal of European Public Policy,* 8(3): 345-370.

Bach, J. (1995). Enough about Process: What we need are heroes. *IEEE Software,* 12(3): 96-98.

Bansler, J. P. and K. Bødker (1993). A Reappraisal of Structured Analysis: Design in an organizational context. *ACM Transactions on Information Systems,* 11(2): 165-193.

Bollinger, T. B. and C. McGowan (1991). A Critical Look at Software Capability Evaluations. *IEEE Software,* 8(4): 25-41.

Brunsson, N. and B. Jacobsson (2000). The Contemporary Expansion of Standardization. In: *A world of standards*. N. Brunsson and B. Jacobsson, editors. Oxford University Press, New York.

CMM Produc Team (2002). Capability Maturity Model Integration (CMMI), version 1.1. CMMI for Software Engineering. Staged Representation. Pittsburgh, PA, Software Engineering Institute.

Curtis, B. (1998). Which Comes First, the Organization or Its Processes? *IEEE Software,* 15(6): 10-13.

Haase, V., R. Messnarz, et al. (1994). Bootstrap: Fine-Tuning Process Assessment. *IEEE Software,* 11(4): 25-35.

Humphrey, W. S. (1990). *Managing the Software Process*. Reading, MA, AFinancialSoftison-Wesley.

IEEE (1998). IEEE Standard for Software Reviews.IEEE Std 1028-1997.

Jacobsson, B. (2000). Standardization and Expert Knowledge. In: *A World of Standards*. N. Brunsson and B. Jacobsson, editors. Oxford University Press, New York.

Kautz, K., S. Madsen and J. Nørbjerg (2007). Persistent Problems and Practices in Information Systems Development. *Information Systems Journal*, 17(3): 217-239.

Madsen, S., K. Kautz and R. Vidgen (2006). A Framework for Understanding How a Unique and Local IS Development Method Emerges in Practice. *European Journal of Information Systems,* 15: 225-238.

Paulk, M. C., B. Curtis, et al. (1993). Capability Maturity Model for Software, v. 1.1. Pitssburgh, PA, Software Engineering Institute, Carnegie Mellon University.

Pries-Heje, J. and I. Aaen (2004). Standardising Software Processes: An obstacle for innovation. In: *IFIP WG 8.6 Conference*. Leixlip, Ireland, Springer.

4

Knowledge Sharing in Software Development

Karlheinz Kautz and Annemette Kjærgaard

1 Introduction

Knowledge management plays a vital role in many professional activities, and software development is no exception here (Kautz & Thaysen 2001). The knowledge management literature is extensive, but the discussion on how to manage knowledge in organisations continues, and new proposals as well as lessons learned are frequently suggested. However, the published literature, especially in the information systems field, is largely grounded in a view that considers knowledge as an objective commodity which can be collected, represented symbolically and processed like information (Dahlbom & Mathiassen 1993; Tsoukas 1998). The literature consequently shows a certain preoccupation with information technology (IT) and technical solutions, while it reflects a limited view of individual and organisational knowledge-related processes (Swan et al. 1999). The practice of knowledge management is too often reduced to the implementation of new IT-based systems, and important organisational aspects, in particular human and social issues, are overlooked.

While studying knowledge management in a Danish software development company, the work presented in this chapter takes these issues into account. Our focus is on the process of knowledge sharing as an important part of knowledge management. Our investigation is driven by the following questions: How does knowledge sharing in software development unfold in practice? What hinders and what helps knowledge sharing in this context? Our interest is primarily in the role of people in the knowledge sharing process, but we do also pay attention to how people use technology to share knowledge.

The chapter is structured as follows. The next section introduces the key concepts which inform our understanding of knowledge sharing. Section 3 presents our research approach and methods. Our empirical findings are described in section 4 and discussed in section 5. The final section presents our conclusions and the challenges for future research.

2 Theoretical Background

We understand knowledge sharing as a bilateral process in which knowledge is exchanged between individuals and groups (Comas & Sieber 2001). Knowledge is the outcome of a complex process, a part of which is the gathering and processing of information. This has been described by Kolb (1984) as a learning process. Learning is significant for the attainment of knowledge, and thus also for the sharing of knowledge. Information communicated among people through a shared language, body language and actions (Fiske 1990; Nielsen 1994), as well as through participation in action and practice, builds the foundation for learning (Wenger 1998). This happens through social interaction and in some cases with the aid of technical media (Thompson & Walsham 2001). Communication and participation in action are thus also significant for the sharing of knowledge. Knowledge, learning, communication and participation in action and their relationship are the pillars of our theoretical framework and are therefore discussed in more detail.

2.1 Knowledge

The definitions of knowledge presented in the literature differ in scope and orientation, but they agree that knowledge is a complex multifaceted concept which can be understood from different perspectives (Cook & Brown 1999; Kautz & Thaysen 2001). From a hermeneutic perspective, knowledge is not a commodity which can be collected under controlled conditions and bought or sold on a market (Dahlbom & Mathiassen 1993). On the contrary, knowledge is subjective enlightenment, a personal property which is grounded in human cognition (Nielsen 1994) and it affords actions (Cook & Brown 1999).

Cook & Brown (1999) apply the concepts of explicit and tacit knowledge. Acknowledging the codifiable nature of explicit knowledge which can be transmitted through formal, systematic language, and tacit knowledge rooted in action and difficult to formalise and communicate, they do not promote one distinct kind of knowledge as an ideal for knowledge. Instead they emphasise the importance and necessity of different forms of knowledge. They distinguish between explicit and tacit knowledge, but also between individual and group knowledge, thereby creating four basic knowledge forms: explicit individual, explicit group, tacit individual and tacit group knowledge. They see these four forms as distinct forms of knowledge, all with equal standing. All four forms are relevant

in describing and understanding knowledge and knowledge sharing in organisations. In addition to the four knowledge forms, the authors introduce the concept of knowing, which is the part of an action or practice which deals with the way knowledge is used in interaction with the social and physical world. Sharing their broader perspective on knowledge, our theoretical framework includes three dimensions of knowledge based on their work: explicit/tacit knowledge, individual/group knowledge and knowledge/knowing.

We thus understand explicit knowledge as knowledge which is codifiable and can be expressed directly through language. This means that explicit knowledge is structured and ordered such that it can be described and discussed through speech, scripts, drawings and other signs and symbols. Tacit knowledge is knowledge which cannot directly be codified, but which can (only) be expressed indirectly through language and action. Tacit knowledge is a 'feel' or sense for something or "to be able to do something without being able to explain why" (Brown & Gray 1995). Tacit knowledge is not hidden or inaccessible – tacit knowledge simply cannot be codified or expressed directly. Explicit and tacit knowledge is intrinsically interrelated and mutually constituted, and tacit knowledge presents the backdrop against which all understanding is distinguished. All knowledge thus has a tacit dimension, and tacit knowledge is the cultural, emotional and cognitive background of which humans are only marginally aware. Explicit and tacit knowledge are inseparably linked; thus, they cannot be treated as two separate types of knowledge (Tsoukas 1996). Against this background, any mechanical conversion model as a background for knowledge sharing such as Nonaka (1994) and Nonaka & Takeuchi's (1995) model of knowledge creation with its conversion processes of socialisation, externalisation, internalisation and combination, appears to be an inadequate description of the underlying processes.

Individual knowledge is knowledge held by an individual and applied in the individual's actions, while group knowledge is knowledge held by a group and applied in the group's actions. Group knowledge is created through the co-operation of the group's members and is part of its practice (Brown & Gray 1995). Group knowledge includes knowledge about how the group works, its social rules, its memory about earlier actions, and knowledge about its tasks. Not necessarily all group members possess the whole group's knowledge; it is the group as a whole that possesses the group's knowledge, and it is the group as a whole that applies this knowledge (Cook & Brown 1999).

Each of the four types of knowledge presented by Cook & Brown (1999) is associated with a set of knowledge forms. Explicit individual knowledge is expressed in concepts, rules, equations and interrelations. This is, for example, knowledge in the form of concepts about the design of a product or a specific procedure to follow. Tacit individual knowledge can be expressed in a person's skills in applying tools or routines for the performance of a particular task (Comas & Sieber 2001). Explicit group knowledge consists of shared stories about previous successes or failures and of metaphors used for establishing a common understanding of a problem or a task. Tacit group knowledge is related to organ-

isational culture in the form of genres which guide the thoughts and actions of organisational members.

The four types of knowledge can be seen as constituting organisational memory in different forms. Organisational memory consists of mental and structural artifacts which have a consequential effect on performance (Walsh & Ungson 1991). In its most basic sense, organisational memory refers to stored information from an organisation's history which can be brought to bear on present decisions.

We have knowledge to carry out an action; at the same time, we can have knowledge independent of whether we carry out the action or not. Knowledge is something we possess irrespective of action. Thus, knowledge is a tool which can be used in action but which in itself is not action. The four forms of knowledge created by the dimensions of explicit-tacit and individual-group are thus not sufficient to describe the knowledge which is expressed through practice (Cook & Brown 1999). Every time someone carries out an action, a combination of many different types of knowledge is used. Each type of knowledge contributes to the action in a way in which the other types cannot. The part of the action which focuses on the application and combination of explicit and tacit knowledge in interaction with the social and physical world is referred to as 'knowing' by Cook & Brown (1999). The authors purport, "knowing is to interact with and honour the world using knowledge as a tool." Knowing has a focus on how we use our knowledge in practice. Knowing is not something we have, but something we do, and therefore a part of the actual action.

2.2 Learning

Learning is the process by which we acquire knowledge based on the communication of information and the participation in action. The learning process is thus an antecedent of having knowledge and is important for the sharing of knowledge.

Kolb (1984) has developed a theory of learning according to which learning is based on experience, and takes place through learning cycles. The learning process is a cognitive process in which individuals process new information or actions based on the background of existing information or actions. In this process information called abstract conceptualisation is obtained though speech and script, or indirectly through action and practice, when it is called concrete experience; thereafter information is processed actively through active experimenting or reflectively through reflection and/or observation. This means that abstract concepts and concrete experience are tested actively in practice or are reflected upon. The result of this process is new or re-organised knowledge.

Comas & Sieber (2001) relate this learning theory to Cook & Brown's understanding of knowledge and knowing and argue that a mutual interplay exists between knowledge and knowing. Knowing means that we use our knowledge in new ways, or that we make use of our knowledge in new ways through practice, and thus discover new interrela-

tions or gain a new understanding of our knowledge. Kolb's (1984) learning cycle, which is based on experiential learning, complements Cook & Brown's (1999) comprehension of knowing, as it can explain those aspects of knowing which deal with the creation of new knowledge. Comas & Sieber (2001) postulate that the four different stages of the learning process indirectly reflect the presence of different states of knowing, and that moving through the cycle contributes to what Cook & Brown (1999) have called productive inquiry. Thus, knowing can be seen as a learning process.

There is, however, a difference between the four types of knowledge in how they relate to learning. The learning process depends on the types of knowledge involved, but also on different ways of learning (Nielsen & Kvale 1999). The concept of scholastic learning describes the part of learning which takes place through verbal and textual instruction detached from practice (Nielsen & Kvale 1999). This comprises the communication of information which represents explicit knowledge and, as such, concepts, rules, equations and interrelations are acquired through scholastic learning. This involves learning through concrete institutionalised education, for example through courses, seminars or literature studies, but also through verbal and textual instruction among colleagues at the workplace.

Learning of tacit knowledge occurs through active participation in practice as well as general learning in day-to-day life. Wenger (1998) presents a social theory of learning in which participation in practice forms the basis for learning. When people work closely together, share a practice, use a common vocabulary to talk about the practice resulting in an understanding of the practice, tacit knowledge in the form of skills, proficiencies, routines and shared genres is more easily shared (Wenger 1998; Wenger & Snyder 2000). Practice learning takes place in communities of practice which are areas of activity or bodies of knowledge which a community has organised itself around. It is a joint enterprise inasmuch as it is understood and continually renegotiated by its members who are linked to each other through their involvement in common activities (Wenger 1998). Over time, a community of practice builds up an agreed set of communal resources, a shared repertoire which consists of tangible as well as intangible aspects such as procedures, politics, rituals and values (Wenger 1998; Wenger et al. 2002).

While scholastic learning is primarily about receiving, processing and absorbing information, practice learning is primarily a question of becoming part of a community (Brown & Gray 1995). Individual learning can occur through scholastic as well as practice learning, and it is an individual's learning of explicit and tacit knowledge. Individual learning denotes the acquisition of knowledge which an individual uses in and for actions. An example is knowledge in the form of concepts about the design of a product, or skills in the application of tools or routines for the performance of particular tasks (Comas & Sieber 2001). Group learning is learning of tacit and explicit knowledge on a group level. Group learning can occur through scholastic as well as practice learning. Through group learning, the group acquires knowledge which it uses for its actions. This comprises

knowledge in the form of rules for how a task is to be solved, sharing of stories of successes and failures, routines for how the group distributes tasks, and genres for particular meetings and documents.

Communities of practice (Lave & Wenger 1991; Wenger 1998) are a prominent example of group learning: groups of individuals who work together over a period of time. The individuals in these groups may carry out the same job or co-operate on a shared assignment, but they do not have to be a formal or identifiable group (Brown & Gray 1995). Individuals in communities of practice are bound together through a shared practice and understanding of this practice, and they develop shared knowledge about the practice (Brown & Gray 1995).

2.3 Communication and Active Participation in Action

The access to and exchange of information is necessary for the acquisition of knowledge (Kautz & Thaysen 2001), and communication is therefore important for the sharing of knowledge. Fiske (1990) argues that communication can be understood as the exchange of information. Fiske relies on Lasswell's (Severin & Tankard 1997)—Who (says) What (to) Whom (in) What Channel (with) What Effect—and Shannon and Weaver's (1949) rather general model of communication, but emphasises that the communication process takes place with the aid of a shared oral and written language, body language or action. In this model a sender or communicator chooses, combines and presents data in such a way that it represents the information which he or she wants to communicate. A receiver recognises and takes information in by interpreting the transmitted data on the background of existing knowledge, other information, context, culture and the rules and pragmatics of the language.

Communication can take place as a dialogue, where the roles of the sender and receiver change continuously during the communication, or as a monologue, where the roles of the sender and the receiver are fixed throughout the communication session. The roles of the sender and receiver of information in the communication process can be held by individuals and by formal or informal groups. Formal groups are groups which are linked to business processes or organisational units and which have been defined by an organisation's management (Brown & Gray 1995). Informal groups are groups which are connected with informal business processes or built by people who congregate outside or across the formal organisational units.

The differences between how individuals or groups exchange knowledge have an impact upon how and which information can be communicated. Information can be found not only in speech, but can also be expressed through body language and action. An individual can obtain information which is embedded in actions and body language by participating in someone else's work. This means that information has different formats, is more

or less structured, and is stored in different ways, which all have an influence on how information can be communicated.

Knowledge sharing takes place through inter-subjective (social interaction) and/or technology-facilitated (technical media) communication (Thompson & Walsham 2001). There is a difference between which type of information can be most suitably exchanged and provided through social interaction and technical media.

Social interaction as the face-to-face exchange of information between people comprises, for example, formal or informal meetings, verbal presentations and teaching sessions. In social interaction, information is communicated through speech, body language or actions; hence social interaction enables the exchange of information linked to both explicit and tacit knowledge (Nielsen & Kvale 1999). Explicit knowledge can be codified, and it can thus be expressed verbally (or textually) in a shared language. Tacit knowledge cannot be codified and it can therefore not be expressed verbally or textually; body language and actions are thus important for the learning of tacit knowledge (Nielsen & Kvale 1999). This indicates that social interaction is suitable for the communication of complex, less structured and non-formalised information because it provides different possibilities for communicating a message. This happens through the application of a combination of a shared language and speech, body language and actions which can supplement each other. Social interaction in itself cannot store information, and thus the storing of the exchanged information through social interaction is dependant on the individuals' or the group's capability to 'store' across time and place.

Technical media are technologies, especially information and communication technologies, which support the exchange of information in the whole or in parts of the communication process. These can be telephones, databases, electronic documents or electronic mails. Technologies can 'transport' communicated information between the involved parties and simultaneously support the storage of data (Walsham & Thomsen 2001). Technology thereby provides the possibility of making information available across time and space. Technology can also contribute to changes in the format or structure of information in the form of categorisation of documents, composition of document indices, or conversion of analogue sound to a data file. This improves the communication in relation to a situation where technology is not applied.

Technical media are suitable for the communication of codified information (Thompson & Walsham 2001) as most are based on verbal and textual communication. Some technologies, such as video, can also reproduce information expressed through body language and action. Technical media are suitable for well structured, well defined and formalised information such as technical specifications or measurement results. Technical media primarily support the communication of information related to explicit knowledge, and they can only indirectly, and to a limited extent, support communication of information related to tacit knowledge (Thompson & Walsham 2001). Different strategies

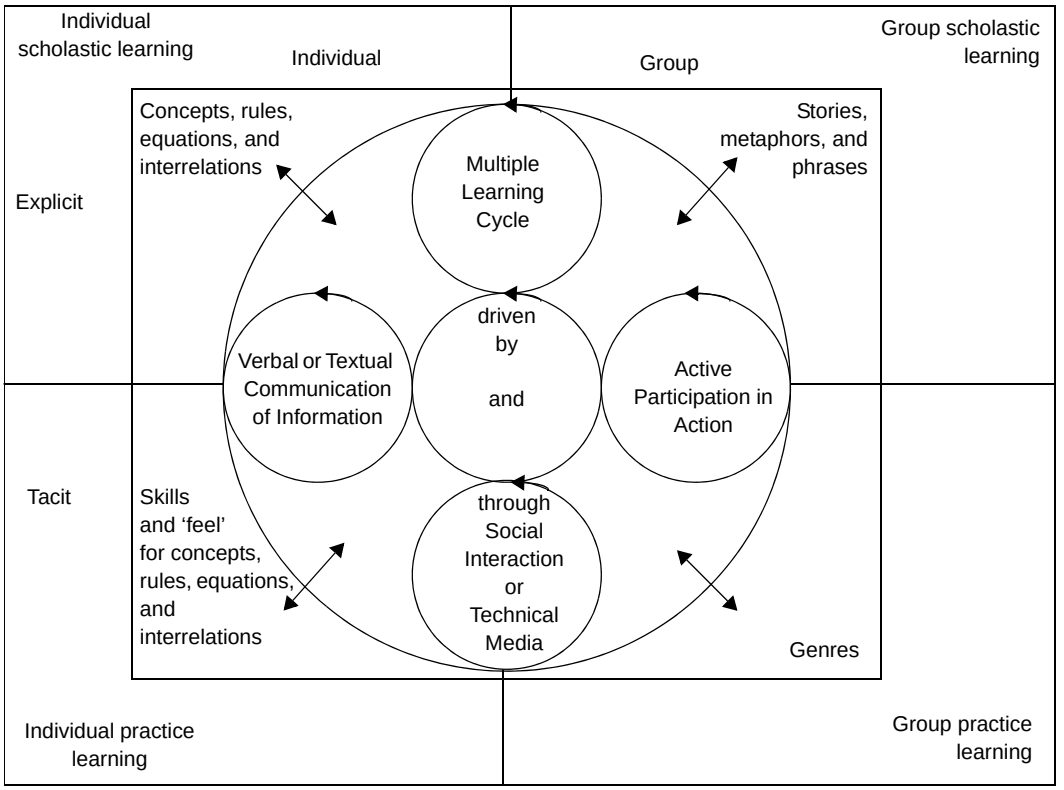

Figure 1: A model of knowledge sharing

are used to access that information depending on the type of knowledge and the background of those who use the medium (Smolik et al. 2005).

In summary, it is our understanding that knowledge, learning, communication and participation in action constitute knowledge sharing via different types of media, see figure 1. Learning occurs through scholastic or practice learning, and the learning process is the process which determines how individuals and groups acquire different types of knowledge through the communication of information and participation in practice. Knowledge, learning, communication and participation in action are related, and they influence each other as prerequisites for knowledge sharing. We will use the framework to analyse the relationship between ways of learning and types of knowledge as prerequisites for knowledge sharing and how knowledge sharing unfolds in our case organisation.

3 Research Approach and Method

The ontological and epistemological assumptions of our work are informed by the interpretive paradigm. The objective of our research was to study and understand the organisational members' knowledge sharing processes within their work and organisational context. In line with Andersen (1999), we chose a case study approach in order to study the process of knowledge sharing in its organisational context.

We concentrated on a single case study of knowledge sharing in a large Danish IT company. This approach might be criticised for only generating a local empirical theory which might not be generalisable, but as argued by Hughes and Jones (2003), it contributes to the existing body of knowledge by providing a detailed account of empirical findings. By limiting the study to a single organisation, we were able to examine the case in more detail to understand more thoroughly the interrelationships of separate data, the essence of our research objective.

We collected data through 13 semi-structured interviews with software developers and project leaders. We also performed one group interview with management and two one-workday-long observations where we followed a manager and a developer. Finally, we included secondary material in the form of different kinds of documents used in the organisation. All interviews were taped, and a summary was written listing relevant issues from each interview. During and after the interviews we produced rich pictures inspired by the Soft Systems Methodology (Checkland & Scholes 1990) to depict our understanding of the situation with regard to knowledge sharing in the organisation. These rich pictures constructed the basis for the data analysis and were so useful that we only had to return to the tape recording for further consultation in one case.

To analyse the data we used a combination of common sense interpretation, inspired by the above rich picture, and theoretical interpretation which related the data to our theoretical framework (Andersen 1999). For all interviews and observations we identified and coded sensible units of statements. These statements were then linked to one or more categories which we deduced from our theoretical framework, and by comparing all statements within each category and across categories, a number of recurring subjects emerged, and over 100 issues critical for knowledge sharing in the case organisation were identified. The main categories, also used in the subsequent in-depth description, were: knowledge sharing through social interaction, knowledge sharing through technical media, and the different kinds and formats of knowledge shared.

This allowed for an in-depth understanding and discussion of knowledge sharing in the case organisation, revealing what kind of interaction and which media provide possibilities and support for sharing different kinds of knowledge. It also revealed what impact the identified issues had on the sharing of knowledge.

4 Knowledge Sharing in the Case Organisation

The case organisation was a Danish subsidiary of a global provider of telecommunication systems. At the time of the study (in 2002), the Danish subsidiary was 23 years old and had been acquired by the American parent company six years earlier. The company's customers were fixed network, mobile network and cable operators residing in more than 80 countries.

The Danish subsidiary focus on the development of new products. The company's products consist of hardware and embedded software which provide the product functionality and management software to access, calibrate and monitor the products' functions. This is mirrored in the organisation of the company's research & development department, which consists of three divisions (access, calibration and monitoring) that perform the detailed specification and actual development tasks. In addition, the company has a product management department, a product planning and specification department and a finance department. The company has 420 employees of whom 180 are directly involved in product development.

We agreed to focus our investigation of the knowledge sharing process on one organisational unit—the division of embedded software consisting of administrative staff, a manager, three projects leaders, two team leaders and 27 developers. With its position between different departments and divisions, embedded software appeared to be the most appropriate and representative for our purpose.

The company was very interested in participating in our study. Due to increased competition in the market, they had decided to focus on employees' knowledge, learning and communication both within and across the organisational units with the aim of increasing productivity and quality in the product development process.

The product development process is organised in programmes consisting of projects. The company has about three large and 5-10 minor programmes running at any one time, consisting of multiple projects of varying length from six months to two years. The projects are typically carried out in and by the different divisions. The company's development process can be characterised as product-oriented, sequential and document-focused. The process consists of an analysis and design phase, a development phase, a test phase, and a product installation and introduction phase.

A planning team consisting of experienced employees from different departments is responsible for the overall definition of a programme. A so-called core group of project managers and technical experts performs a feasibility study and produces product and document reviews at the end of each development phase. Finally, a core team consisting of the project managers from the different divisions in R & D is responsible for conducting and completing the actual projects and products. The development work is performed by the individual divisions and their developers and is co-ordinated by the project managers in and across the three divisions and beyond.

The developers in the division were our primary unit of investigation. Within the category 'individuals and groups' we differentiated between developers in the division, developers who worked on the same tasks, developers who shared a two-staff office, project groups, sub-project groups, informal groups, and experience (exchange) groups. Individuals and groups related to other units were roughly distinguished into: 1) individuals in other divisions and departments affiliated with the same project or product 2) other departments and divisions in the Danish subsidiary, or 3) individuals and groups in other countries.

The case study showed that the software developers drew upon organisational memory to share knowledge. Knowledge sharing as it unfolds through social interaction took place in the case organisation through formal interaction in formal meetings, informal interaction in informal meetings, seminars and courses, presentations, exchange of experience groups, personal networks, and in the offices shared by two or more people. Knowledge sharing as it unfolds through technical media, happened through the use of the organisation's document handling system (DHS), its file server, error reporting system, version control system, project management system, electronic mail, electronic discussion groups, internet, intranet, video, whiteboards, paper documents and literature. The kinds of knowledge that the developers shared covered technical specifications (particularly requirement specifications and design documents), general project information and technical standards, information about who knows what about previous projects or old products, and information about technical problems and improvement of products.

In the next four subsections we provide a more detailed analysis of how knowledge sharing unfolded between individuals as well as within and between groups and of the types of knowledge shared. We also identify critical issues related to each of the categories and provide some suggestions regarding how some of these issues can be solved. As limitations of space preclude a full description of our analysis, we have chosen two subjects from each of the categories to illustrate our findings.

4.1 Knowledge Sharing through Social Interaction: Formal meetings and Personal Networks

The developers and managers participate in various formal meetings dealing with the communication of the status of projects and problems through verbal and textual communication. These formal meetings provide the opportunity for the acquisition of explicit individual and group knowledge. Explicit individual knowledge is shared in the form of coordination, communication of status and discussions of tasks. Explicit group knowledge is shared in the form of (communication of) management attitudes and opinions, definition of rules and (communication of) stories about good and bad projects. The formal meetings are also used as 'pathfinders' to those developers who hold individual knowledge about, for example, concepts and rules, or who have specific skills. In this way

the formal meetings provide the individuals with the possibility of obtaining knowledge about where they can obtain further knowledge. Active participation in formal meetings also provides the opportunity for acquisition of knowledge about the genres which are related to formal meetings and the acquisition of individual skills about how meetings are run.

However, the information communicated through project meetings is arbitrary. The developers are not sure that the information they need, or which would be helpful to them, is communicated through meetings. This contributes to the project meetings having less value for the developers, and indicates that the developers are not sure about the genres which are related to formal meetings, or that they lack skills to utilise formal meetings. The study does not provide any evidence that the developers are trained in useing formal meetings. The developers therefore communicate to a higher degree through informal meetings which are held as the need for particular information arises, or through personal networks. Information about the same subject is communicated through many different forms of social interaction and through different technical media. This in itself does not have to be a problem, as a certain amount of redundancy increases the chance that everyone will receive the desired or necessary information, but it is problematic when it confuses the developers.

Furthermore, there are no project meetings between divisions, and coordination across divisions rarely happens in formal meetings. Communication between the divisions is generally poor, and is limited to taking place through the project managers' personal networks. This reinforces the organisational divide between the developers in the different divisions and creates a barrier which does not support communication or insights into other developers' work. In practice, the communication and the exchanges, which happen across the departments and are necessary for the developers, come about through personal networks or written information. The communication between divisions is much more difficult than within a division because one has to be much more precise, as the developers cannot assume that the receiver of their information has the same background knowledge.

Personal networks are personal relationships between the developers. They are used to communicate information and to instruct each other. Communication of information through personal networks is verbal and textual, and provides the opportunity for scholastic learning of explicit knowledge. This is true for both individual and explicit group knowledge because personal networks are directed towards communication of concrete problems, discussions of different possibilities for solutions, and the informal education of the individual developer. This knowledge takes the form of concepts and rules related to projects, products and other technical matters which the developers need for the performance of their work. Good and bad experiences are communicated for this purpose, and phrases are used which are defined by the groups in the organisation. This happens, for example, through informal meetings, which are the links for the networks and also make them more visible. The developers communicate information through informal meetings;

at the same time, they can identify other developers' personal networks, thereby building and extending their own personal networks.

The developers become actively involved and participate in each others' practice through personal networks, and in so doing they have the opportunity to acquire the genres which are related to personal networks. In addition, the networks give the developers the opportunity to acquire skills in how personal networks can be used, and skills related to the topics which are communicated. This happens, for example, through shared offices, which facilitate the developers' participation in the same practice. It is common practice in the company that an experienced developer shares an office with a less experienced developer to help him or her to achieve skills and learn genres. This includes helping the less experienced by pointing in the right direction when it comes to searching information. Areas in which this strategy occurs are in communication about how things work, where specific information can be found, as well as in the form of stories and rules related to work in the divisions.

The poorer the formal meetings and the technical specifications, the more the developers use their personal networks. Personal networks are pivotal for communication in the division because many other forms of social interaction and technical media are not adequate and detailed enough for the developers to communicate all the desired or necessary information. The developers gain a sense of this through the personal network for what is important and whom they should go to in order to prioritise resources and tasks. Personal networks bind other forms of social interaction and technical media together, and they satisfy the need for communication which is not covered through other social interactions and technical media.

Developers depend on personal networks to be able to perform the work for which they need information from previous projects. These networks are strongest within the division, but more experienced developers also have networks which span divisions. This dependency on the personal networks and on the most experienced developers can be problematic when the developers who have comprehensive knowledge of previous projects and old products leave the organisation and take their knowledge with them.

Personal networks save time. They are faster to use than searching in documents. The developers can find what they are looking for faster through their personal networks than through other forms of social interaction or technical media. Personal networks are created through the developers' daily work, but a good network takes many years to build up. This is because the developers work on the same projects and with the same colleagues over a long period of time. This contributes to the situation where some developers have broad and comprehensive networks with a lot of information at their disposal, while others have lesser networks have less information. This indicates that not all developers have comprehensive personal networks, which results in less communication of information and fewer opportunities for the acquisition of new knowledge for these developers.

The analysis of the examples of social interaction shows that social interaction through verbal and textual communication provides the possibility for scholastic learning of explicit knowledge. Beyond this, the active participation in action in the form of social interaction provides the opportunity to acquire those genres which are related to social interaction and the acquisition of the individual skills for how social interaction can be applied, and the skills related to the topics and themes which are communicated. The investigation shows that the individuals and groups in the investigated division communicate through social interaction, and social interaction thus provides the opportunity to share knowledge in the division. This is interesting, as the analysis also shows that social interaction is not facilitated in a structured way, and that formal forms of social interaction in general are not prioritised. This creates problems, as the developers are uncertain about which information is communicated through which type of social interaction, resulting in a high degree of arbitrariness with respect to which information is communicated through the form of social interaction: in courses, presentations, experience groups or personal networks.

4.2 Knowledge Sharing Through Technical Media: The document handling system and the file server

The document handling system (DHS) provides the opportunity for textual communication. By far the largest part of all technical specifications, analysis documents and project-related documents is stored in the DHS. This comprises feature plans, functional and object models and iteration plans, but also how-to documents, technical standards and presentations. The documents are used by the individual developers in their work as they describe what and how something should be or has been done. The documents are also directed towards groups as they provide the framework and the scope for the groups' collective work. The verbal and textual communication through the DHS offers the opportunity for individuals and groups to acquire explicit knowledge.

Active participation in the form of interaction with and use of the DHS provides the possibility for the developers to gain skills in use of the system and in acquiring the genres related it. This comprises which documents are stored in the system, how the documents are structured and categorised, how search functions are applied, and how keywords and indices are linked to documents.

Not all documents are stored in the DHS and, as a consequence, not all necessary documents are accessible to all developers. The DHS is regarded as being for documents which are directly related to projects or the product development process. Other documents which do not comprise formal or project-defining information, such as descriptions of 'personal' ideas for solutions, experience or general descriptions of problems, are often stored on other media, e.g., the file server. Only what the individual developer thinks has to be version-controlled is stored in the DHS. The developers have to search for informa-

tion in many places and to supplement information from the DHS with information from, e.g., the file server or their personal networks. This indicates that developers need to have a good overview of the information and have to make an extra effort to gather the quantity and the quality of information that they need for their work.

The DHS is slow and hard to use, the search functions are poor, and distribution lists are not used to the necessary degree. It is difficult to find the right documents, there is a lack of abbreviations and designations, and the information structure in the system is strictly hierarchical. The actual functionality and the information structure in the DHS hinder the developers' use of the DHS and the communication of information. This limits the developers' chances of sighting all documents and finding all relevant information for a particular task. The strict information structure does not mirror the individuals' and the groups' problem-oriented way of working. Communication of information in the DHS is based on the partition between projects and divisions instead of being based on the structure of the working processes, problems and solutions. In addition, not all departments are equally engaged in using the system. This reinforces and increases the problems related to the developers' limited possibilities of accessing all documents and finding all relevant information for a particular task. Although the DHS is the official place for storing and distributing documents, some departments use other technical media, for example, databases for technical specifications.

Many developers use the system only to the extent to which it has an impact on their communication with other divisions, or when demanded as part of the formal work process. Some divisions provide their documents to developers in other divisions only to a very limited extent, and the division also only has an insight into other divisions' documents to a limited extent. The lack of insight into each others' work makes the communication of information across divisions difficult and hinders the sharing and exploitation of explicit knowledge. In the case where the use of the DHS is bypassed to the advantage of other technical media, this also has an influence on the acquisition of tacit knowledge related to the use of the DHS, especially the individual skills related to using the DHS for the storage of technical specifications. This is also true of the acquisition of genres related to the DHS. Information about genres is communicated though the documents in the DHS. When the same information is communicated through different forms of social interaction and technical media, meaning that different genres are used for the same themes, this can lead to a situation where the same information gains different meanings.

The file server is primarily used for informal documents such as technical notes, proposals for technical designs, intermediate and temporary versions of documents, standards, product descriptions, third party documents, etc., which are not directly related to the projects or the product development process. Apart from these documents, the file server also contains binary files in the form of programs and video recordings. This shows that the file server is used for textual and verbal—in the form of video recordings—communication, which provides the opportunity for acquisition of explicit knowledge related

to the information which is stored on the server. Interaction with and use of the file server provides the opportunity for the developers to acquire skills in the use of the file server and genres related to it. These are implicitly determined by the developers and deal, for example, with which documents and files are to be stored on the server, how they are and should be structured, and which search functions can and should be applied.

The file server is used to store files which should have been stored in the DHS. A problem arises when the DHS is regarded as the storage means for documents which are directly linked to the projects or to the product development process. Nevertheless, some older documents have not been transferred to the DHS and are stored on the server. This includes, for example, own ideas and experiences, technical notes, proposals for technical designs, designs, intermediate and temporary versions of documents, standards, product descriptions, third party documents etc. which are not directly or indirectly related to the projects or the product under development. This indicates that information which relates to the same subject is stored and communicated through different technical media: for example, own experience and proposals for technical designs are stored on the file server, while technical specifications related to the same product are stored in and communicated through the DHS. This results in a situation where the developers have to gather information from different places, depending on whether the information is categorised as formal or informal, or whether the information relates to newer or older products. Thus, developers must have a good overview of the information and require extra time to gather the necessary quantity and quality of information. As information has an impact on the learning process and ultimately the acquisition of knowledge, the use of the file server hinders the sharing of explicit knowledge, since it really functions as a ragbag and provides the opportunity to store information in places other than the DHS.

The developers' and the divisions' use of the file server is very different. Some developers never use it; others use it often or store and find much useful information there. The divisions also structure files differently on the server, and this is an obstacle to the sharing of knowledge. The different use of the file server makes the communication of information through this technical medium look arbitrary and dependant on individuals. The different use of the file server holds the risk that the developers do not communicate the same information, and thus that the information accessible to the other developers becomes different. There is a risk that those developers who do not use or do not have knowledge of the information on the file server do not acquire knowledge of the same concepts or stories. This then decreases their later capabilities to act. The developers' different uses of the file server lead to differences in the extent to which the developers acquire skills related to the file server and knowledge of the file server as a genre. This includes skills related to the use of the file server to store and work with technical standards or the handling of video recordings or other binary files on the server. It also includes the genres related to the file server, for example, the form in which information in a technical standard is communicated through the medium.

It is difficult to find information on the file server. The developers have to know where the information is stored to find it. The problem with access to information on the file server is emphasised by this issue. This is due to the fact that the file server has limited search functions which only cover searching based on file names, and that the directory structures and naming are very 'anarchistic.' This means that the developers have to have considerable knowledge of the file server and its contents to be able to search for information there. Information about the file server is communicated through personal networks, but the personal networks take a long time to build up and seldom cross division boundaries. It is thus difficult for new employees and developers in other divisions to find relevant information on the server.

In summary, all these issues deal with access to information. As information is the basis of the developers' acquisition of knowledge, access to and gathering of information are necessary for the acquisition and sharing of knowledge. The analysis shows that the issues identified in relation to the developers' use of technical media to communicate have a negative influence on the sharing of knowledge between individuals and groups in the company division which was investigated.

4.3 Sharing Different Kinds of Knowledge: Technical specifications and information about 'who knows what'

Technical specifications comprise feature plans, overview and detailed function and object specifications and hardware specifications. They are the basis for the acquisition of explicit knowledge, and the developers actually use particular technical specifications for product development. The technical specifications are directed towards the individual developer's work and at the same time define the rules and phrases for shared development work. They thereby form the basis for the sharing of explicit knowledge between individuals as well as groups.

The developers, who have been in the division for a long time, can understand and use technical specifications better than others. This indicates that through their daily work, the developers acquire skills and genres in utilising technical specifications. Active participation in action provides the opportunity for the acquisition of tacit knowledge. This means that the developers' use of technical specifications contributes to their skills in utilising technical specifications. Their skills related to the topics which the technical specifications deal with, and to the acquisition of genres which are related to technical specifications.

However, not all developers understand the documents equally well. When they do not understand the documents they contact the developers who originally produced the documents and ask them what they were trying to describe and express. This shows that the developers need certain skills and knowledge of the applied genres to be able to use the information which is provided in the technical specification.

The developers find it difficult to gain an overview of what has been agreed on, and which of the changes in the specifications have been carried out. The changes are sometimes carried out before they are agreed upon and approved, and not all changes are documented and reviewed. These problems show that communication about changes or updates to specifications and documentation is sometimes difficult or defective. Acquisition of knowledge is based on the information which the receiver obtains. Access to and exchange of information is thus necessary for the acquisition of knowledge. Therefore, the developers' different and non-updated information basis has an impact on the explicit knowledge that they acquire, and ultimately on their possibilities of action.

The change management tool is not used to receive information on updated technical specifications, and the document handling system is sometimes bypassed and e-mail is used instead. Communication of these documents and information on updated technical specifications takes place through many channels and different technical media. This indicates that the developers are not proficient in and certain about the genres which are connected to these technical media, e.g., the specifications are communicated through the document management system (DHS), but updates to these documents are communicated through the error reporting system, while both types of information should be communicated through the DHS.

Sometimes feature plans are not completed when a project starts, resulting in guess work and rough estimates, and there is no possibility of tracing all features through the whole product development process, including the final test of the product. As a consequence the quality of the information which the developers use in their work is insufficient. As noted earlier, access to and exchange of information is necessary for the acquisition of knowledge, and when information is not available, acquisition of knowledge is impeded or hindered.

Information on 'who knows what' includes information about which colleagues have the skills and knowledge about projects, products and technologies. As this information is relevant to the individual developer's seeking of information and simultaneously defines rules for who the developer should turn to, this information is the basis for explicit individual and group knowledge.

Active participation in action provides the possibility of acquisition of tacit knowledge. This means that the developers' utilisation of personalised knowledge contributes to: (1) utilising these skills and finding the colleagues in question, and (2) the acquisition of the genres which are related to the skills and to these people. The study shows that the developers use a lot of time finding out who knows what, and that they do not always succeed. The developers find that knowledge about who knows what comes with time, when they become familiar with other developers.

Many developers lack insight into who has what information and skills. The developers often only know the skills of those they are working with or have worked with in projects. Many developers have limited knowledge of their associates. However, for devel-

opment of the technical skills it is important to know whom to ask in the organisation. The lack of knowledge of colleagues can thus have a negative impact on the access to and communication of information, and ultimately a negative impact on knowledge sharing, as the developers do not have an overview of what information can be found where. This is even more problematic as it is most important to know whom to ask as part of the development of the skills which the developers require to perform their tasks.

Project managers are prominent as they know who has what knowledge. However, the overview over who has what information and which skills, and therefore the access to other people's information and capabilities, depends on a relatively small number of people. The division's catalogue of expertise is no longer updated, and therefore it is of very limited help in this respect. The developers instead use their personal networks to obtain this information. The difficulties in localising information and skills of other developers are reinforced through the fact that the developers have difficulty finding overviews of who works in which project. Thus, important or necessary information cannot be found and the developers do not actually know what others need to know. This is a great hindrance to the distribution of information to developers who require it.

In summary, the analysis of technical specifications and information about 'who knows what' as exemplars of different kinds of knowledge shows that the identified issues can be detrimental to the sharing of knowledge in the organisation, as this knowledge is difficult to find, and may also be hard to understand. The developers instead use their personal networks to acquire the knowledge that hey need. However, as personal networks take time to build, this solution works better for those developers who have been with the company for a longer time.

4.4 Sharing Different Types of Knowledge: The relationships between explicit individual, explicit group, tacit individual, tacit group knowledge

A closer look to our empirical data also reveals how the relationships between explicit individual, explicit group, tacit individual, tacit group knowledge contribute to the acquisition and sharing of knowledge.

With regard to the relationship between individual and group knowledge a number of developers in our study explained that they discussed technical problems and solutions with other developers using, personal networks and experience groups. We found that the developers had a shared repertoire of phrases for particular types of technical specification. Thus, the developers' individual explicit knowledge about how technological solutions are specified contributes to the group's use of phrases concerning technical specifications, and therefore to explicit group knowledge about specific types of technical specifications.

We also found that the developers primarily store formal documents, i.e., documents which are a defined part of the development process or which are somehow related to projects and products in the DHS. These documents are placed in a strict hierarchical structure which is ordered according to departments, products and/or projects. This indicates that the developers' skills—tacit individual knowledge of skills and a 'feel' for which documents should be stored in the DHS, and where—contributes to the tacit group knowledge of a genre with respect to which documents are communicated through the system and how they are communicated.

The study also shows that new developers are trained in the use of the DHS through courses and apprenticeship learning. This indicates that the tacit group knowledge of a genre related to the DHS contributes to the development of the individual developer's skills in relation to the use of these genres.

Regarding the relationship between explicit and tacit knowledge, a number of the developers reported that on the basis of books, presentations, and courses, they use their understanding of concepts and interrelations to experiment in practice and thereby acquire new skills. The individual developers apply explicit knowledge in practice—practice learning which contributes to the acquisition of tacit knowledge in the form of skills. The study also shows that developers who are experts in a particular field tell other developers about how a specific task should be carried out. This can happen through presentations, experience groups and personal networks, thereby giving other developers the opportunity to gain explicit knowledge about the action related to solving the task in question. This indicates that the developers do reflect upon their tacit knowledge and explicate related actions and activities, which they then communicate to other developers through scholastic learning.

The study also shows that the project management system which was previously used to plan projects is now only used for general project planning and time registration. According to the respondents, it ought to be possible to extract estimates for new projects directly from the system; however, because the projects are dissimilar, these estimates are hard to re-use. We also found that some developers had ceased to register time in the system. This indicates that stories, i.e., explicit group knowledge, about the deficiencies of the project management system make the developers interpret the information which is communicated through the system in an unfavourable way. They have thus developed a perception of the system which has become tacit group knowledge. The study also shows that the developers reflect over their use of the DHS as a form of communication in the company which only functions poorly. This contributes to stories among the developers about how their work is made cumbersome by the system, and that necessary documents are hard to find. This shows that the groups reflect on their tacit knowledge and explicate actions related to this knowledge which contribute to the acquisition of stories and phrases in the groups related to the use of genres.

Concerning the relationships between tacit individual and explicit group knowledge and explicit individual and tacit group knowledge, one developer, through use of his skills, developed a browser tool which other developers found useful. It became part of the development toolbox and contributed to the stories and phrases which the members of this group of developers told about useful tools. This shows that developers, through their skills and thus their individual tacit knowledge and its explication in related actions, contribute to the group's stories and phrases, and thus their explicit group knowledge.

The study also provided examples of how success stories of solving specific tasks were sometimes transformed into how-to documents, for example, how to program in C, which other developers could use to acquire new skills. This indicates that a group's stories enter into individual practice learning which contributes to the development of the individual group member's skills, in this case programming in C.

The developers' understanding of the importance of standards for the development work makes them perceive what is communicated as a standard as being specifically important. This indicates that concepts or rules related to the use of a particular form of communication contribute to the development of genres. The study also shows that when developers observe others communicating, they gain knowledge about how, for example, one conducts a presentation or a formal meeting. This indicates that the developers' acquaintance and application of genres contribute to the developers' individual understandings of the concepts, rules and interrelations which are related to genres.

5 Discussion

The study has shown that our framework can explain the relationship between the communication of information, participation in action, the learning process, and knowing the different types of knowledge. It presents an appropriate basis for a detailed understanding of knowledge sharing. The empirical data provide examples of how different forms of knowledge are shared in social interaction and by using different media.

Our study shows how knowledge sharing unfolds and that when there are problems with the communication of information, such as missing information, poor access to information or poor quality of information, this determines the extent to which explicit knowledge can be acquired. Similarly, if active participation and learning in practice are hampered or functioning poorly, this determines the extent to which tacit knowledge can be acquired.

Our work is based on the assumption that a number of different dimensions and perspectives have to be taken into account in order to create a comprehensive understanding of the knowledge sharing process. Other authors only deal with the relationship between two dimensions, such as communication of information and knowledge, or knowledge and learning, or they consider and describe the dimensions as different perspectives on knowledge sharing. We take all these different dimensions and perspectives into account.

Cook & Brown (1999) identify types of knowledge and knowing which can exist in an organisation and the relationship between them. They consider knowing primarily as productive inquiry. They relate them neither to the concept of learning nor to the importance of communication of information. Comas & Sieber (2001) discuss managing knowing by describing the relationship between experiential learning and knowing. They argue that managing knowing can be described as an experiential process, and they thus emphasise knowing primarily as a learning process. They do not, however, draw on the importance of what precedes this process, namely, communication of information and the actual situation in which learning takes place. Kolb's (1984) learning cycle concentrates primarily on the learning process with a focus on individual learning which ignores the group dimension of learning and knowledge. Kolb (1984) also emphasises how information is gathered and processed in the learning process, but does not deal with what happens before in the form of communication of information. He thus ignores the problems which can arise when obtaining information. Nielsen & Kvale (1999) identify scholastic learning and practice learning, as well as the explicit knowledge and tacit knowledge to which these ways of learning contribute, but they do not clearly deal with the individual and group dimensions of knowledge and learning. We take all these issues into account. Our understanding of communication is distinct from the original models (Fiske 1990) by being explicitly directed towards our focus on knowledge and learning processes, and by viewing communication not as a technical, de-contextualised process, but as a social process which takes place in a context. We thus emphasise the concepts of social interaction and technical media instead of the technical term 'channel', and we do not use the terms 'sender' and 'receiver', but stress the concepts 'individual' and 'group' in relation to communication. Our theoretical framework thus contributes to research within the field of knowledge sharing by integrating different areas related to knowledge sharing.

We have seen how different types of knowledge contribute to the learning of different types of knowledge, and how the developers constantly use knowledge to learn in order to create new knowledge. These learning situations can also be understood as multiple cycles which can take place simultaneously. According to Cook & Brown (1999), learning is a process in which an individual or a group processes new information on the basis of existing knowledge which can then be used as a basis for action. The result of the learning process is new knowledge. Following Kolb's (1984) understanding of the learning process and the relationship between different ways of learning and different types of knowledge as expressed in our framework, and relating and comparing these to the relationship of different types of knowledge, the following becomes evident:

Tacit knowledge is acquired through practice learning, but when, for example, the developers explain how a task is solved, they reflect upon tacit knowledge. The situation where a developer explains to other developers what he or she has done is scholastic learning. It is not the tacit knowledge itself which becomes explicit through scholastic learning, but the developers who acquire new explicit knowledge. This is not the same as being able

to solve a particular task. If other developers want to achieve the same tacit knowledge, this can only happen through practice, i.e., they themselves must apply the explicit knowledge in practice and participate actively in carrying out the task. Tacit knowledge and explicit knowledge supplement each other through reflection and application in practice. Explicit knowledge can contribute to the acquisition of tacit knowledge through application in practice. At the same time, tacit knowledge contributes to the acquisition of explicit knowledge through reflection and participation in multiple cycles of action.

Explicit knowledge can contribute to the acquisition of tacit knowledge through application in practice and through participation in practice learning. At the same time, tacit knowledge contributes to the acquisition of explicit knowledge through reflection and participation in scholastic learning. This can be understood as a detailed description of knowing as a learning process. Taking scholastic learning as a starting point, explicit knowledge will always precede an eventual acquisition of tacit knowledge. Taking practice learning as a starting point, tacit knowledge will always precede an eventual acquisition of explicit knowledge. It is a question of whether one starts with scholastic or with practice learning.

As a result, the theoretical framework can be presented in a model of sharing knowledge, where communication consists of verbal and textual communication and active participation in action. Learning comprises learning through scholastic learning, practice learning and multiple learning cycles, and knowledge consists of explicit and tacit individual and explicit and tacit group knowledge.

The model for the sharing of knowledge shows that verbal and textual communication of information through social interaction and/or technical media contribute to individual and group scholastic learning, which thereby contribute to explicit individual and group knowledge. The model also shows that participation in action through social interaction and/or technical media contributes to individual and group practice learning, which thereby contribute to tacit individual and group knowledge. In addition, the model for the sharing of knowledge shows how multiple learning cycles and knowing contribute to the acquisition of knowledge.

6 Conclusions

In the research presented here, we have investigated how and which information is shared in a software development organisation through communication via social interaction or different types of technical media.

Our work resulted in a model of knowledge sharing which could be used to understand and ultimately improve knowledge sharing in practice. Although some concrete advice for improving knowledge sharing could be derived from our analysis, the theoretical framework and our study do not describe how the results of the analysis and the identified areas for improvements could be transformed into concrete organisational or technical

improvements, or which strategies organisations should be used use to improve the sharing of knowledge.

Problems related to the technical media are related to limited access to and quality of information. The developers in the case organisation use or supplement them with social interaction or other technical media which enable them to gain access to the information or actions which are a prerequisite for their learning processes, and the acquisition of knowledge which is necessary for them to carry out their tasks. How an optimal balance between social interaction and technical media might look has to be decided from case to case. This could be constructed with the aid of Hansen et al.'s (1999) work on codification and personalisation strategies. In this context, it is necessary for future research to investigate the relationship between the different types of knowledge in more detail in order to clarify how this relationship can be used to improve the sharing of knowledge.

Finally, the study shows that personal networks play a specific role in the sharing of knowledge, as they link other forms of communication together and compensate for the information which is not communicated through these other forms of communication. The framework does not explain all phenomena related to personal networks with regard to knowledge sharing, and future research based on the framework should thus aim to extend the framework.

References

Andersen, I. (1999). *The Apparent Reality—On knowledge production in the social sciences*. (in Danish), third edition, Samfundslitteratur, Copenhagen, Denmark.

Brown, J. S. and E. S. Gray (1995). *The people are the Company*, http://www.fastcompany.com/magazine/01/people.html (accessed 01.11.2005).

Checkland, P. and J. Scholes (1990). *Soft Systems Methodology in Action*. Toronto, Canada., John Wiley and Sons.

Comas, J. and S. Sieber (2001). Connecting Knowledge Management and Experiential Learning to Gain New Insights and Research Perspectives. In: *Proceeding of the ECIS 2001*, Bled, Slowenia.

Cook, S. D. N. and J. S. Brown (1999). Bridging Epistemologies: The generative dance between organisational knowledge and organisational knowing. *Organisational Science*, (4): 381-400.

Dahlbom, B. and L. Mathiassen (1993). *Computers in Context—The philosophy and practice of systems design*. Cambridge, UK, Blackwell.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review*, 14(4): 532-550.

Fiske, J. (1990). *Introduction to Communication Studies*. Second edition, London, UK, Routlegde.

Hansen, M. T., N. Nohria and T. Tierney (1999). What is Your Strategy for Managing Knowledge?. *Harvard Business Review*, March-April: 106-116.

Hughes, J. and S. Jones (2003), Reflections on the Use of Grounded Theory in Interpretive Information Systems Research. In: *Proceedings of the ECIS 2003 Conference*, Naples, Italy.

Kautz, K. and K. Thaysen (2001). Knowledge, Learning and IT Support in a Small Software Company, *Journal of Knowledge Management*, 5(4): 349-357.

Klein, H. and M. Myers (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. *MIS Quarterly*, 23, (1): 67-93.

Kolb, D. A. (1984). *Experiential Learning—Experience as Source of Learning and Development*. New Jersey, USA, Prentice Hall.

Lave, J. and E. Wenger (1991). *Situated Learning: Legitimate peripheral participation*. Cambridge, UK, Cambridge University Press.

Nielsen, K. and S. Kvale (1999). *Master-Apprenticeship Learning: Learning as social practice*. Copenhagen, Denmark, Hans Reitzel (in Danish).

Nonaka, I. (1994). A Dynamic Theory of Organisational Knowledge Creation. *Organisation Science*, 5 (1): 14-37.

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

Polanyi, M. (1966), *The Tacit Dimension*. London, UK, Routlegde.

Severin, W. and J. Tankard (1997). *Communication Theories*. Fourth edition. New York: Longman, 1997.

Shannon, C. and W. Weaver (1949). *The Mathematical Theory of Communication*. Urbana, University of Illinois Press.

Smolnik, S., S. Kremer and L. Kolbe (2005). Continuum of Context Explication: Knowledge discovery through process-oriented portals. *International Journal of Knowledge Management,* 1(1): 27-46.

Stenmark, D. (2000). Leveraging Tact organisational Knowledge. *Journal of Management Information Systems.* 17(3): 9-24.

Swan, J., H. Scarbrough and J. Preston (1999). Knowledge Management – The next fad to forget people. In: *Proceedings of the ECIS 1999*, Copenhagen, Denmark.

Thompson, M. and G. Walsham (2001). Learning to Value the Bardi Tradition: Culture, communication, and organisational knowledge. In: *Proceeding of the ECIS 2001,* Bled, Slowenia.

Tsoukas, H. (1998). The Word and the World: A critique of representationalism in management research, *International Journal of Public Administration*, 21(5): 781-817.

Tsoukas, H. (1996). The Firm as a Distributed Knowledge System: A constructionist approach. *Strategic Management Journal*, 7(Winter Special Issue): 11-25.

Walsh, J. P. and G. R. Ungson (1991). Organisational Memory. *Academy of Management Review,* 16**:** 57-91.

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

Wenger, E. C. and W. M. Snyder (2000). *Communities of Practice: The organisational frontier.* Harvard Business Review, 78:139-145.

Wenger, E. C., R. McDermott and W. M. Snyder (2002). *Communities of Practice – A guide to managing knowledge.* Boston, Harvard Business School Press.

Part II

Techniques

5

Making Sense of Project Management

Annemette Kjærgaard, Karlheinz Kautz and Peter Axel Nielsen

1 Introduction

Managing knowledge in software development has been of interest to IS researchers for quite a while. Examples are the experience factory (Basili and Green 1994; Rus and Lindvall 2002), the application of knowledge management theory to explain software process improvement (Arent and Nørbjerg 2000; Baskerville and Pries-Heje 1999; Kautz et al. 2002; Mathiassen and Pourkomeylian 2003), and the role of knowledge in software firms (Agarwal and Prasad 2000; Althoff et al. 2000). Nevertheless, the discussion on how to manage knowledge in software development is far from finished and new proposals, as well as lessons learned, are continually being suggested. It remains a problem, though, that published research in the information systems field in general views knowledge as an objective commodity which can be collected, represented symbolically and processed like information (Dahlbom and Mathiassen 1993; Tsoukas 1998). As a consequence, the literature shows a certain preoccupation with information technology and technical solutions, while it reflects a limited view of individual and organizational knowledge related processes (Swan et al. 1999). The practice of knowledge management is often reduced to the implementation of new IT-based systems, procedures for documenting and sharing information, and the documents themselves though there are examples to the contrary. By focusing on externalisation and documentation of knowledge, important organizational aspects, in particular human and social issues, can be overlooked.

This chapter takes these issues into account and is based on a perspective of knowledge and knowledge management as an ongoing sensemaking process (Weick 1995). Our focus is on the process of creating a shared handbook for project management in software development. By using sensemaking as the analytical lens, we explore how software developers make sense of the process of managing projects, and how this influences their perception of what should be included in a shared handbook.

Based upon an empirical study of this knowledge management process in a Danish software development company, we provide insight into how developers make sense of project management based on their construed realities (Isabella 1990). We discuss the value and relevance of taking a sensemaking perspective and suggest further use thereof. The question which guided our research can be summarized as: What general understandings of project management do the participants in the improvement project draw from when they contribute to the handbook, and how do they make sense of its content and use?

This chapter is structured as follows. The next section presents our research approach and methodology. The theoretical framework, which we use to analyse our empirical findings, is described in section three, and the case company is introduced in section four. Section five reports the analysis of the findings. Section six discusses the findings and contributes suggestions for future research. The final section addresses conclusions.

2 Research Approach and Methodology

The research presented in this paper is interpretive. It is based on an empirical case study and action research in SpaceSoft, a Danish software company, where the development of a project handbook was started as part of a larger project focused on software process improvement. The project handbook development project was conducted in order to improve SpaceSoft's project management practices and to support knowledge sharing amongst the project managers. The project ran for more than 12 months with the first version of the handbook presented after eight months. Our research comprises this time period, but also includes a larger contextual study and other improvement projects.

The study is process-focused (Pettigrew 1997) where we study the process of developing a project handbook rather than the project handbook itself. The study adopts a process perspective in which the process of developing the handbook is defined as "a sequence of individual and collective events, actions and activities unfolding over time in context" (Pettigrew 1997, p. 338).

The roles and length of stay in the field have varied for the three authors of this paper. One author was involved in the project as an action researcher throughout the three-year period (Iversen et al. 2004; Mathiassen 2002; McKay and Marshall 2001). This author was involved in development of the handbook from the very early stages of the project, although he did not contribute to the actual writing process of the handbook. A second researcher participated as an action case (Braa and Vidgen 1999) or involved researcher

(Walsham 1995) for three months, contributing primarily to the interpretation of the case as sensemaking. A third researcher acted as an outside observer of the case study (Walsham 1995) and primarily contributed to the analysis of the data. The combination of intervention, interpretation and collaboration between the three researchers with different levels of involvement was chosen to bring interpretive rigour to the project and to introduce new analytical perspectives on the case (Madsen et al. 2006). This research design counters the criticism of action research which suggests it can be easily reduced to mere consultancy (Baskerville and Wood-Harper 1996).

2.1 Data Collection

The project handbook development process was part of a research collaboration project. It was undertaken by several project managers appointed by the CEO, the CEO himself and three action researchers.

Data was collected through active participation in the development process as well as through written documents.

Active participation took place during meetings held at regular intervals, typically once a month. During these meetings the action researchers captured data by participating and observing, while at the same time taking notes. Data was also collected at informal meetings and in corridor talks which enabled a fuller understanding of what was discussed at the meetings, as well as of different participants' opinions about the process. These were important insights that we would not have gained from exclusive participation in the formal meetings.

A variety of *written documents* were collected, such as: the original project proposal, drafts of the handbook, email correspondence, minutes of meetings, company documents, as well as project reports and deliverables. These documents provided vital information of a more formal nature not only on which decisions had been taken and what the background was and plans were, but also how these changed over time during the project.

2.2 Data Analysis

In line with the interpretive approach, the analysis of the project handbook development process came about through an iterative process of interpretation and comparison of the data, based on the use of the analytical lens of sensemaking (Weick 1995); this forms the basis for the theoretical framework (Kjærgaard and Kautz 2008) to be presented in the following section. The analysis was conducted in two stages.

The first stage took place during the process where the participating researchers continuously analysed the development process and used this analysis to intervene and to suggest new actions. Throughout the process, the participating researchers discussed their observations and possible interventions in the process in order to construct a more coher-

ent view of the development. In cases of misunderstandings of the process, the researchers actively sought to solve issues or clarify inconsistencies in perceptions.

In the second stage of the analysis, we explicitly sought to map the sensemaking process of the case onto the process model of knowledge management as organisational sensemaking (presented in section 3). Every discrepancy between the case and the model in this initial mapping was discussed at length and in such detail that it could be resolved. In every instance where a significant interpretation of the case did not immediately fit well into the framework, great care was taken not to force the case and our interpretation onto the framework, which is presented next.

3 A Process Model of Knowledge Management as Organisational Sensemaking

To analyse the data, we use a process model developed in an earlier process study of establishing organizational knowledge management (Kjærgaard and Kautz 2008). The model was constructed to understand knowledge management as an organizational sensemaking process unfolding in the interplay between collective understandings of the organisation and individual members' meaning construction. In this study we use the model to explore the sensemaking processes which were in play in the process of constructing the project management handbook at SpaceSoft.

The model explains knowledge management as a sensemaking process unfolding in the interaction between collective and individual cognition over time.

The model operates with two sub-processes of the knowledge management process, *creating* and *negotiating*, each of which has a dominant *construed reality* as well as a set of *cognitive processes*. Further, a triggering event of *collision* between expectations and experiences marks the transition from one process to the other. Although there is no specific indicator for time, the change from one stage to another indicates that the process unfolds over time.

The process model is shown in figure 1. The top layer of the model shows the organisational members' dominant frame of reference at the two stages of the process: creating and negotiating. These construed realities (Isabella 1990) are collectively held perceptions of how to behave in the organization. A construed reality is an assembly of more or less connected pieces of information and beliefs, which together form a picture that confirms or constructs a reality (Isabella 1990).

From a sensemaking perspective, the construed reality can be seen as constituting the frame which forms part of the meaning construction equation in sensemaking. The frame in sensemaking is described as an overall paradigm or shared understanding to which cues are related to create meaning (Weick 1995). Frames tend to be past moments of socialisation which relate present moments of experience (cues) to create meaning. According to

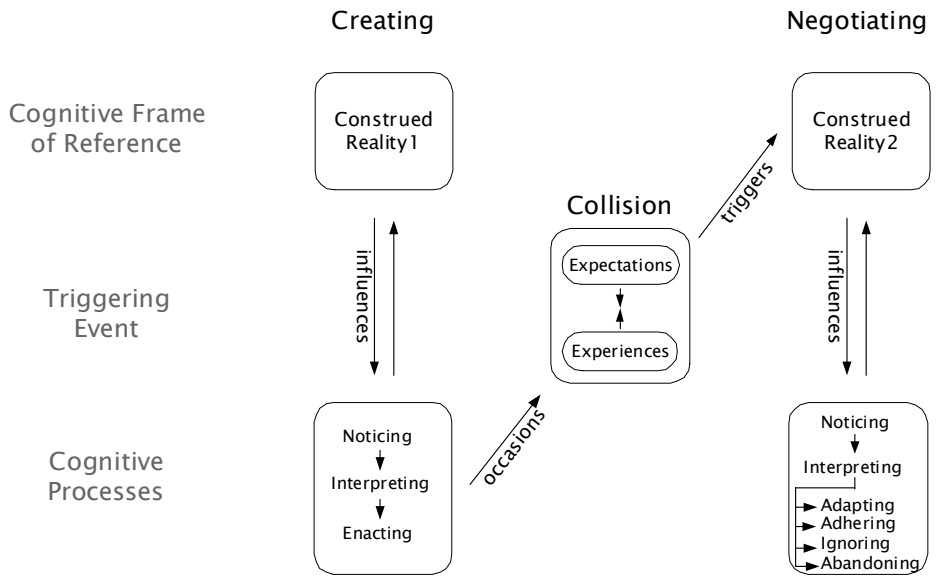

Figure 1: Basic flow of the process model of knowledge management as processes of sensemaking, from (Kjærgaard and Kautz 2008)

Weick (1995, p. 111), meaning is created if a person can construct a relation between past and present moments, "The content of sensemaking is to be found in the frames and categories that summarise past experience, in the cues and labels that snare specifics of present experience, and in the ways these two settings of experience are connected."

In the model only one construed reality is shown as the cognitive frame in each of the sub-processes of creating and negotiating. However, more construed realities can co-exist (Pratt and Foreman 2000; Smircich and Stubbart 1985). What we refer to in the model is therefore the dominant construed reality in each sub-process.

The bottom layer of the model shows the cognitive processes at the individual level, i.e., the cognitive processes which form the organisational members' sensemaking and meaning construction. The arrows from the construed realities to the cognitive processes show that the individual processes are influenced by the collectively held understandings of the organization.

The cognitive processes in the first period of creating are based on the understanding that members notice particular parts of the ongoing flow of information that they are exposed to continuously. Weick explains this as belief-driven sensemaking which is the effort to create meaningful action based on 'expecting.' According to Weick, beliefs are obvious anchors in organisational sensemaking and they are found in ideologies, cultures, scripts, and traditions (Weick 1995, p. 155). Expectations tend to be very directional in their operations and severely filter the input, thus influencing what is being noticed. Ex-

pectations are building blocks for the construction of an object which is subsequently noticed as real. According to Weick, "...[w]hen perceivers act on their expectations, they may enact what they predict will be there. And when they see what they have enacted, using their predictions as a lens, they often confirm their prediction" (Weick 1995, p. 152). In this respect, sensemaking is as much about plausibility and coherence as it is about accuracy.

What organisational members bracket out of the ongoing flow in the creating process is based on the dominant construed reality which guides their *noticing* (Weick 1995). Following this, they *interpret* the noticed information drawn from the construed reality, which they then *enact*. The arrows from the cognitive processes pointing back to the construed reality show that the process of enactment confirms and strengthens the construed reality. This cyclic process can be seen as a process of thinking in circles or a self-fulfilling prophecy (Weick 1979; Weick 1995).

Intermediating the two sub-processes of the knowledge management process and the two layers of cognitive processes, is a collision between expectations and experiences. It is the members' perception of dissonance between expectations and experiences which creates uncertainty and triggers the members to pursue stability by establishing a new frame into which their experiences fit. In sensemaking terms this collision between expectations and experiences can be referred to as an occasion for sensemaking or a shock (Weick 1995) which triggers a more intense process of meaning construction in order for the members to reinterpret the dominant construed reality or to create a new??

Although sensemaking is a continuous process, it can intensify in two types of occasions which can cause a 'shock', and thereby initiate an instance of sensemaking in organisations: ambiguity and uncertainty (Weick 1995, p. 91). Both types of occasion create an interruption in an ongoing flow, although the 'shock' is different in the two types. Ambiguity is the situation where "the assumptions necessary for rational decision making are not met" (Weick 1995, p. 92). The problem here is not that the information is insufficient, but that more information may not resolve the misunderstandings. In layman's terms, ambiguity may be referred to as confusion. Uncertainty, however, governs when there is a lack of knowledge, and it might thus be resolved by gaining additional information. Simply said, uncertainty can be referred to as ignorance.

Basically any kind of experience can serve as an occasion for sensemaking as long as it is unexpected and creates a gap between the way things are and the way they 'should have been' (Hansen 2002). Some occasions initiate unconscious sensemaking (automatic sensemaking) and some initiate conscious sensemaking. In both cases the important point is to understand what happens in the sensemaking process which might have implications for the subsequent meaning construction which guides further action and sensemaking of a situation.

The cognitive processes in the second period of negotiating are more ambiguous than those of the first period as they are influenced by a new construed reality which dominates

but has not necessarily replaced the construed reality of the creating process. Whereas sensemaking in the creating process is driven by beliefs in the form of expecting and arguing, sensemaking in the negotiating process is based on action as well as beliefs. The action which guides sensemaking in the negotiation process is primarily of the members' own creation (sensemaking as committing). The basic idea behind sensemaking as committing is that by committing an action, the action is made irrevocable and thus more difficult to change than the beliefs about that action. Commitment thus "imposes a form of logic on the interpretation of action" (Weick 1995, p. 159). Commitment, on the one hand, reduces flexibility, learning and adaptation because it makes withdrawal difficult; thus, commitment slows adaptation to change. On the other hand, commitment makes it easier to get things done, as it focuses the social construction of reality on those actions which are high in choice, visibility and irrevocability (Weick 1995, p. 162). By creating the initiatives on the basis of the first dominantly construed reality in the creating process, the members commit themselves to this action. In the negotiating process, this commitment brings the members to search for explanations to make sense of their own actions.

Depending on which of the construed realities dominate their sensemaking process, the members create meaning out of their experiences and subsequently act upon this meaning. The cognitive processes in the process of negotiating thus reflect the extent to which the actors adapt, adhere, ignore or abandon their actions.

One possible reaction is that members *adapt* their beliefs to the new construed reality and consequently adjust their activities to better fit the new situation. Adaptation in this situation is a classic learning situation where members' experience becomes encoded into the construed reality as a realisation of the incongruence between expectations and experience. Learning can be seen as having taken place by the updating of their view of reality.

Alternatively, members *adhere* to the construed reality dominating the creating process. Whereas the process of creating is based on members' expectations and thus has beliefs as the driving substance of the sensemaking process, a dominant element of the negotiating process is the actions themselves. In other words, members make sense of a situation that they have created themselves, resulting in what Weick (1995) refers to as an action-driven sensemaking process of committing to one's own actions.

A third response is to *ignore* experiences and simply try again. Where the processes of adapting and adhering both entail continuation of the action generated in the creating process, the process of ignoring entails discontinuation of this action. Members make sense of their lack of success by blaming their own inadequacy to create good enough initiatives for management to accept. Thereby they still draw from construed reality, but discontinue the action.

Finally, members may accept the new reality and *abandon* the action, acknowledging that it does not fit the new construed reality. By abandoning the action, members accept

that the organisation has changed and that the previously proposed ideas do not fit the organisation any longer.

In the next section the empirical findings from our study are analysed using the framework of knowledge management as sensemaking.

4 Case Description: SpaceSoft

We carried out our empirical studies in the company, SpaceSoft, which develops software as a subcontractor to the European Space Agency (ESA). SpaceSoft develops a wide range of dedicated software for on-board, micro-gravity, verification and validation, ground station control, and check-out systems.

The company, founded in 1992, is a rather old company in the software development business, and it has lived through many changes. Most of the company's software developers have a M.Sc. in engineering or computer science. A considerable number of software developers have developed software for the space industry for many years and are quite experienced with the particulars of space products. An equal number of software developers have little experience within the space industry, but rely on experience from other domains of software development.

In 2002 SpaceSoft decided to focus on improving its software processes and entered into a collaborative research project called Software Processes and Knowledge. The initiative to improve software processes started with a traditional, although light-weight, assessment of the current software processes, based on the understanding of the action researchers that improvement should be initiated with an assessment, (Mathiassen et al. 2002) albeit not necessarily with a formal and model-based assessment (Iversen et al. 1999). SpaceSoft's current software processes were compared informally to the processes in the Bootstrap model (Kuvaja 1999; Kuvaja et al. 1994). The result indicated significant discrepancies between the Bootstrap model and the company's current software processes and practices. This led to a decision to prioritize improvement of requirements engineering and project management although other processes were also in need of improvement. The focus of this paper is on how SpaceSoft addressed improvement of project management.

The European Space Agency has numerous standards that its subcontractors, including SpaceSoft, must comply with. A number of these are process standards, and SpaceSoft has in the past dealt with the issues of compliance uniquely in each development project. The ESA standards are quite complex. They form a hierarchy of standards, with the levels of details in instructions varying considerably, as well as whether instructions are required, recommended, or optional. Thus, the documentation of compliance is never trivial and it requires project managers to be well-read in the many ESA standards. This led to the first idea for improvement, namely, that a new and improved project management process should be documented in a *handbook*. The handbook needed to be compliant with the rel-

evant ESA standards; it was expected that it would be much easier to document the compliance when project management was performed in accordance with the handbook guidelines.

It is SpaceSoft's declared strategy to deliver fixed-priced software on time. The CEO, in particular, expressed a deep concern for achieving this goal and was very clear in maintaining that all improvement activities should be directed at this. There was a strong belief among some of the experienced project managers and the CEO that SpaceSoft was already applying a number of *best practices*, i.e., established practices within the discipline and within the industry. This belief was also confirmed by SpaceSoft already being in compliance with the ESA standards, as these standards would then inherently reflect the best practices within the software space industry. This led to the idea that the software processes for project management should encompass these already existing practices. Thus, although they already knew what to do, it was just not documented in a shared handbook.

Other project managers questioned the need for improvement; they wondered whether their industry-best practices just needed to be enacted by project managers, thus eliminating the need for improvement. Their argument was that not all project managers had the necessary training in these processes; given this lack of competence, some *training and education* was necessary, especially for new project managers. This led to the idea that a project manager education programme should be established, based on the new and improved processes.

The improvement project started with a series of workshops with the purpose of: 1) designing and writing a handbook for project management and 2) designing a training course for the use of the handbook. In the workshops, three to five experienced project managers participated, together with the CEO and two to three action researchers.

During the first workshop, the (sub)processes of project management were identified. This identification was based mainly on the project managers' experience and secondarily on the ESA standards. Some of these processes were easily written by one to two project managers and they never led to any controversy in later workshops. A few significant processes were very difficult for the project managers to write. Although the design goal for the handbook was to be brief, they eventually drafted quite complex process descriptions, which were then heavily criticised by other project managers at later workshops. This led to much iteration over the process descriptions where sections were expanded and condensed several times, while the core ideas of the processes were constantly negotiated during the workshops.

The handbook was designed, written and reviewed in these workshops, but the aims of the workshops changed during the progress of the project, with some attention directed to related problems. The training course was never designed, and the handbook gradually expanded its scope to become a handbook for software engineering processes as well. The workshops were never chaotic, but there was disagreement about various issues. Most

disagreements were overcome and problems were solved. However, a fundamental issue arose and gradually caught the attention of the participants during the last workshops. This issue was a concern for where the project managers' knowledge came from and how they could share this knowledge.

This issue arose from the participants' experience that there were two groups of processes: (a) the 'easy processes' which were easy because the project managers and the CEO shared the knowledge necessary to perform and describe the processes, and (b) the 'difficult processes' which were difficult because the participants did not share the necessary knowledge for writing the procedures. A few of the project managers and the CEO in particular were of the opinion that the handbook should be written and formalised, and that all project managers should then perform the formalised processes based on the instructions in the handbook. Other project managers held the view that the handbook should be more than a set of instructions; it should also contain explanations that would enable project managers to perform the processes.

To analyse these differences in more detail, we use a framework of knowledge management as sensemaking.

5 Analysis

The theoretical framework, presented in section three, enabled us to analyse our data from a knowledge management perspective, focusing on how participants made sense of the handbook as a means for process improvement.

In the first period of creating the handbook, two differently construed realities were at play. First, there was the construed reality of the CEO in which he viewed the project management processes as mere descriptions that project managers were to follow rigorously. This construed reality became apparent during the workshops when we heard the CEO speak of why he wanted a handbook developed in the first place. Earlier in his career, the CEO had been an experienced project manager and, based on his own experiences, he challenged the project managers to manage projects in a much stricter way than they had previously done. His argument was that "if he could do it, they could as well." The CEO's construed reality bracketed his view of project management in SpaceSoft and guided what he noticed and interpreted and subsequently enacted. As a consequence, his contribution to the handbook reflected an emphasis on what plans and procedures should be followed in the project management process.

Second, the project managers generally shared a construed reality deeply rooted in their day-to-day experience with project management. Included in this experience were past projects which had been late, changing requirements negotiated with ESA managers, and plain difficulties of delivering products at the agreed time with the agreed features. To these project managers it did not make sense to simply stick to plans and follow procedures. Whenever problems in projects arose they rarely experienced alleviation of the

problems by adhering to the plan or by re-planning. To them, the problems were usually unanticipated, for example, they may have unknowingly allowed the ESA manager to introduce the trouble, or a technical issue proved to be a much bigger risk late in the project. These project managers believed in planning, but only to a limited extent. In their noticing and interpreting of events and problems, they relied therefore only to a limited extent on plans and procedures, because a much stricter process did not make sense to them. What they enacted, when constructing the handbook, was therefore a much more flexible view of project management as a constantly changing process, which could not be controlled by plans and procedures.

During the first workshops both of these construed realities influenced the discussions and the development of the project management handbook. The different ways of interpreting and enacting the contents of the handbook were based on different construed realities which led to much confusion in discussions that could not immediately be reconciled. Gradually, it became apparent that in the common wish to develop the handbook, participants experienced a *collision* between how the project managers and the CEO perceived project management in software development. The CEO expected the project managers to be able to avoid project overruns and maintain the initially negotiated budget and thus contribute to profitability of the company as a whole. The CEO and the project managers believed from experience that there had been too many incidents in the past where these expectations had not been fulfilled. However, the project managers found the CEO's expectations to be less realistic and thought that he was not dealing with the kind of trouble that they were facing.

Only gradually did this collision lead the workshop participants to realize that the views embedded in the project management handbook needed to be negotiated. During the *negotiation process* the CEO and the project managers generally came to share a construed reality in which there was a sense that the project managers already had the competence to perform most of a project manager's tasks. However, it was also acknowledged that sometimes the CEO had to insist upon contracts and agreements to be adhered to, and that plans were agreements between a project manager and the CEO. Moreover, an understanding emerged that project managers needed to use all possible means (e.g., written documentation, plans, checklists, procedures) to remember all important and relevant issues because forgetfulness and lack of anticipation were bound to lead to troubled projects. This negotiation of 'reality' is illustrated by three examples in the PM handbook: a technique for estimation, a template for a project plan, and a procedure for shipment, described in greater detail below.

5.1 A Technique for Estimation

A huge debate arose when the first draft of how to conduct estimation in projects was reviewed. The draft included several important concepts of estimation, gave a number of

techniques and ended by promoting a 3-point estimation, which was then explained in some detail. The draft was written with the intention of explaining estimation techniques also for those that had not previously seen nor applied estimation techniques. At that time the most common way for project managers to estimate was guessing based on the project manager's past experience, but there was significant variation in how the project managers estimated projects and tasks. The draft was briefer than a standard textbook on software engineering, but it covered similar content. During the discussion there was a strong feeling that these techniques would not improve the estimation accuracy, but also that the draft was either too detailed for those who already knew the techniques or too abstract for those who did not. After a long discussion the author of that draft section was persuaded to rewrite it.

At a later workshop the second draft was discussed. That draft contained only the recommended 3-point estimation and a brief terminology on different estimates used by both ESA and SpaceSoft. It was evident that for new project managers the PM handbook could not substitute training in estimation, but it did provide the necessary common ground and it was compliant with ESA standards.

At the final approval of this section of the handbook the construed reality had reached a state in the project managers' perception where they were very likely to adhere to this part of the process.

5.2 A Template for a Project Plan

The project plans differed significantly. ESA did not require a particular way, nor did SpaceSoft internally. Most of the differences between project plans could not be explained by the difference between the projects' conditions and content. There was a genuine desire among the project managers to formulate a template that would remind the project managers of what was important in a project plan. This was strongly supported by the CEO. The CEO particularly wanted the project plan to be viewed as a contract between him and the project manager. They should go through the details in the project plan before it was approved and it should be monitored and discussed at regular meetings between the CEO and the project manager. It was important that there was agreement on what the contents of a project plan should be. After this was clarified during the second workshop, it was relatively easy for the author of this section of the handbook to rewrite it, and it was later approved.

The template in its final form would stipulate headings and contents in the project plan, but it would also stipulate the stages the plan needed to go through (e.g., draft, approved) and how the project manager and the CEO would discuss the monitoring of progress. By the end of the last workshop there was agreement among the project managers that the template was highly relevant and that they would begin using it. It was very

general and in every instance where it was to be used it had to be adapted to suit the specific needs of the project.

5.3 A Procedure for Shipment

There had been several incidents in the past where projects had not been able to deliver products to the customer on time or with items missing due to troubles with packaging and shipment. A procedure was suggested by the CEO that would ensure, in detail, that all possible problems in the final shipping were taken care of in due time. The procedure included very detailed instructions, e.g., on shipping of hardware in wooden boxes if necessary, on customs clearance and on assembly manuals. At first the project managers found it much too detailed. It was in sharp contrast to other parts of the handbook where many details had been left out. After much discussion it was evident that the procedure was really to be viewed as a checklist. A checklist had the advantage of reminding a very busy or otherwise forgetful project manager of small, but crucial issues to deal with in due time.

The difference between the procedure for shipment and the technique for estimation is that shipment is a routine task, but it can be harmed by forgetting just a small part. Estimation, on the other hand, is not routine and appropriate estimation relies on the project managers' competence and experience. It cannot become routine to the same degree. After the realization that the handbook could easily contain several types of processes—some highly dependent on competence and some highly routinized—the project managers agreed to adhere to the procedure.

6 Discussion

The handbook consists of elements varying from procedures to be followed to general guidelines that require the project managers to possess considerable knowledge and competence in advance.

Our case shows that the view that the project management handbook is seen as an externalisation of the project managers' knowledge about project management is too simplistic. While the project managers' knowledge does indeed contribute to the handbook, this knowledge is far from unequivocal and in some cases it clashes with the knowledge of the CEO. The case shows that the participants in the project drew from different construed realities of what project management should be and had been in the past. These differently construed realities pointed to what they believed to be central or important about project management and thus what should be included in the handbook. In other words, they did not see what the others saw as important about project management, and it is doubtful whether the handbook would have been of any use to them, if they had not been through a process of collective sensemaking to create a shared construed reality, giv-

ing sense to the content of the handbook (Gioia and Chittipeddi 1991). The findings from the case suggest that the sensemaking process is a vital part of the process of constructing a handbook. By analysing how the participants' sensemaking processes unfolded, a more complex understanding was created of how the participants perceived the handbook and, as a result, what they suggested it should contain.

Another related issue, which became apparent in the study, was the consequence of a predominant rational understanding of project management combined with a likewise predominant rational understanding of knowledge management. Project management and also software process improvement in software engineering is often presented through a very instrumental view on knowledge management. Knowledge management is taken to be a question of what should be stored in a database and how the data in the base should be searched (Althoff et al. 2000; Conradi and Dingsoyr 2000; Lindvall et al. 2001). There has already been some critique voiced on this view based on a theoretical stance (Aaen et al. 1998). For example, our case study shows from an empirical stance that the handbook as a storage of knowledge is very limited and misses the significance of the project managers' experiences and competencies. The same can be said about knowledge management where a technical-rational understanding has also dominated (Swan et al. 1999). The consequence of this was also apparent in our study where the rational approach to project management, primarily promoted by the CEO, led to an overly rational approach to knowledge management. Knowledge management was initially limited to focus on the process of externalising the participants' knowledge into written procedures and thus brought about a rather simplistic view of what forms of knowledge are important in project management. It also led to a rather ineffective knowledge creation process. However, triggered by the collision of this approach with the project managers' experiences, the negotiation process brought to light a more complex reality. This reality was subsequently externalised to a certain extent, but, as explained by Tsoukas (1996), acknowledging that explicit and tacit knowledge is intrinsically interrelated and mutually constituted, and that all knowledge thus has a tacit dimension, it more importantly becomes part of the group's tacit knowledge.

Also, some implications for practice can be drawn from the study. Our empirical findings point to the issue of who takes ownership of the handbook and its contents. Ideally, the CEO and the project managers together should take responsibility and this is often assumed to be the case in organizations which decide to create some form of knowledge repository. However, our analysis shows that ownership has to be negotiated between the contributors to the handbook in a process of collective sensemaking. In other words, our findings indicate that the negotiation process is itself important for making clear what understanding of reality guides the contributors' contributions to the handbook. By focusing on the process and the ongoing negotiation, a much more complex understanding of what is knowledge emerges.

7 Conclusion

The case and the analysis illustrate a set of complex knowledge processes that the framework has helped us untangle. The types are briefly summarised as the creation and negotiation of knowledge. Where creation of knowledge initially relied on how the participants made sense of project management based on their own construed realities, the negotiation of knowledge invoked a renewed sensemaking process. In the negotiation process the participants reconsidered their construed realities and constructed a new shared construed reality based on social interaction, which helped them to make sense of the various approaches to project management, which were held among the participants of the group.

In conclusion, our findings show that by applying a sensemaking framework, we gain a better understanding of the complexity involved in creating a shared object for knowledge management in software development. Whereas the dominant view in the IS literature on knowledge management still focuses on the externalization of knowledge, the sensemaking framework enables us to understand what general understanding the participants draw from when they contribute to the creation of organizational knowledge. The sensemaking framework and the analysis of our case may have crucial effect on software development and software process improvement. The conclusion is that an improvement effort needs to go through the two processes of creating and negotiating.

References

Agarwal, R. and J. Prasad (2000). A Field Study of the Adoption of Software Process Innovations by Information Systems Professionals. *IEEE Transactions On Engineering Management*, 47(3): 295-308.

Althoff, K.-D., F. Bomarius and C. Tautz (2000). Knowledge Management for Building Learning Software Organizations. *Information Systems Frontiers*, 2(3/4): 349-367.

Arent, J. and J. Nørbjerg (2000). Software Process Improvement as Organizational Knowledge Creation: a multiple case analysis. In: *Proceedings of the 33rd Hawaii International Conference on System Sciences*. Wailea, Hawaii.

Basili, V. and S. Green (1994). Software Process Evolution at the SEL. *IEEE Software*, 11(4): 58-66.

Baskerville, R. and J. Pries-Heje (1999). Knowledge Capability and Maturity in Software Management. *The DATABASE for Advances in Information Systems*, 30(2).

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

Braa, K. and R. Vidgen (1999). Interpretation, Intervention and Reduction in the Organizatioanl Laboratory: A framework for in-context information systems research. *Accounting, Management and Information Technologies*, 9: 25-47.

Conradi, R. and T. Dingsoyr (2000). Software Experience Bases: A consolidated evaluation and status report. In: *Product Focused Software Process Improvement*. 1840, pp. 391-406.

Dahlbom, B. and L. Mathiassen (1993). *Computers in Context - The Philosophy and Practice of Systems Design*. Cambridge, UK, Blackwell.

Gioia, D. A. and K. Chittipeddi (1991). Sensemaking and Sensegiving in Strategic Change Initiation. *Strategic Management Journal* 12(6): 443-448.

Isabella, L. A. (1990). Evolving Interpretations as a Change Unfolds: How managers construe key organisational events. *Academy of Management Journal*, 33(1): 7-41.

Iversen, J., L. Mathiassen and P. A. Nielsen (2004). Managing Risk in Software Process Improvement: An action research approach. *MIS Quarterly*, 28(3): 395-433.

Iversen, J., P. A. Nielsen and J. Nørbjerg (1999). Situated Assessment of Problems in Software Development. *The DATABASE for Advances in Information Systems*, 30(2): 66-81.

Kautz, K. H., K. Thaysen and M. T. Vendelø (2002). Knowledge Creation and IT Systems in a Small Software Firm. *Journal of OR Insight*.

Kjærgaard, A. and K. Kautz (2008). A Process Model of Establishing Knowledge Management: Insights from a longitudinal field study. *Omega*, 36(2): 282-297.

Kuvaja, P. (1999). BOOTSTRAP 3.0—A SPICE1 Conformant Software Process Assessment Methodology. *Software Quality Journal,*(8): 7-19.

Kuvaja, P., J. Similä, L. Krzanik, A. Bicego, S. Saukkonen and G. Koch (1994). *Software Process Assessment & Improvement - The Bootstrap Approach*, Blackwell Publisher.

Lindvall, M., M. Frey, P. Costa and R. Tesoriero (2001). Lessons Learned about Structuring and Describing Experience for Three Experience Bases. In: *Learning Software Organizations (LSO)*. K.-D. Althoff, R. L. Feldmann and W. Müller, eds, Springer Lecture Notes in Computer Science 2176.

Madsen, S., K. Kautz and R. Vidgen (2006). A Framework for Understanding How a Unique and Local IS Development Method Emerges in Practice. *European Journal of Information Systems*, 15(2): 225–238.

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

Mathiassen, L. & P. Pourkomeylian, (2003). Managing knowledge in a software organization. *Journal of Knowledge Management*, 7(2): 63–80.

Mathiassen, L., J. Pries-Heje and O. Ngwenyama, eds. (2002). *Improving Software Organizations: From principles to practice*. Reading, MA, Addison-Wesley.

McKay, J. and P. Marshall (2001). The Dual Imperatives of Action Research. *Information Technology & People*, 14(1): 46-59.

Pettigrew, A. M. (1997). What is a Processual Analysis? *Scandinavian Journal of Management*, 3(4): 337-348.

Pratt, M., G. and P. O. Foreman (2000). Classifying Managerial Responses to Multiple Organizational Identities. *Academy of Management Review*, 25(1): 18-42.

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

Smircich, L. and C. Stubbart (1985). Strategic Management in an Enacted World. *Academy of Management Review*, 10(4): 724-736.

Swan, J., H. Scarbrough and J. Preston (1999). KM - The Next Fad to Forget People. In: *Proceedings of the 7th European Conference on Information Systems*. J. Pries-Heje, C. U. Ciborra, K. Kautz, J. Valor, E. Christiaanse, D. Avison and C. Heje, eds., Copenhagen, Denmark.

Tsoukas, H. (1998). The Word and the World: A critique of representationalism in management research. *International Journal of Public Administration*, 21(5): 781-817.

Walsham, G. (1995). Interpretive Case Studies in IS Research: Nature and
method. *European Journal of Information Systems*, 4: 74–81.

Weick, K. E. (1979). *The Social Psychology of Organizing*. New York, McGraw-Hill.

Weick, K. E. (1995). *Sensemaking in Organizations*. Thousand Oaks, CA, Sage.

Aaen, I., P. Bøttcher and L. Mathiassen (1998). The Software Factory: Contributions and illusions. In: *6th European Conference on Information Systems (ECIS)*. W. R. J. Baets, ed., Aix en Provence, France.

6

Mapping Knowledge Flows

Karlheinz Kautz and Bo Hansen Hansen

1 Introduction

The theme of this book is the importance of knowledge for the improvement of software development. Software process improvement (SPI) is concerned with strengthening software development capabilities, which for us means that one primary task of SPI research is to strengthen the knowledge managing and sharing capabilities of software development companies (Hansen and Kautz 2004).

With this background, this chapter presents a means for identifying and analyzing how knowledge flows through an organization. We have developed this technique for mapping knowledge flows between organizational actors and assessed its applicability in a pilot experiment in Systematic Software Engineering (SSE), see also chapter 11. In the experiment we tested and evaluated one way in which the technique could be used to analyze an organization's knowledge sharing capabilities; thus, it points to opportunities for improvement and provides recommendations for how it can support SPI based on knowledge management.

The chapter is organized as follows. The next section presents the research background and approach. In section 3 the mapping techniques is introduced and section 4 includes the description of the pilot application and experiment of the technique. The experiment is briefly discussed in section 5 after which conclusions are provided.

2 Research Background and Approach

The starting point for the development of the technique was the insight that improving knowledge management and knowledge sharing leads to improved software development practices, see chapter 4**.** We were therefore looking for a technique to analyze how knowledge is shared, to determine which sharing mechanisms work well and how to resolve the problems that result from knowledge sharing. The term knowledge flow was coined to express the belief that knowledge that is shared by many individuals beyond close working relationships in different organizational units and on different organizational levels "flows through an organization" (Hansen and Kautz 2005).

The development and test of the technique took place during a ten month period. In a pilot experiment the concept, technique and application were developed in close collaboration between researchers and the employees of the Danish software company SSE. It followed a research approach inspired by collaborative practice research (CPR) (Mathiassen 2002). The CPR approach is based on a three step repeatable process. First, the understanding of a subject area is achieved through collection and interpretation of data, in our case the insight that we both lacked an appropriate understanding of the organization's knowledge sharing capabilities and an appropriate means to achieve such an understanding. Second, on this basis, support is designed which, in our case, led to the technique for mapping the knowledge flow (further described in section 3). Third, through intervention the designed support means are applied to prove their value and to enhance practice. For this purpose, we performed an experiment with the technique (see section 4). As input to the experiment we collected data from eight semi-structured interviews with employees from all organizational levels. Further, we analyzed artifacts which were used by SSE, such as templates for reports, manuals describing organizational processes, computer based tools, etc. In addition, one of the authors was present in SSE once a week and participated in the daily routines. This gave him a deeper understanding of the organization and its culture, which also served as input to the mapping experiment.

In CPR the outcome of such a research effort can be used as a basis for a new understanding and then trigger further research cycles. This was also the case here (see subsection 4.3 and Hansen 2009).

3 The Mapping Technique

Mapping techniques have long been used to analyze problem areas which have not been understood by different stakeholders. In this context, maps are defined as being "an interpretive description of a situation" (Lanzara and Mathiassen 1985) and are, as such, an interpreted model of reality; that is, a map consists of selections of relevant details of the mapped situation and provides information about what the mapmaker(s) find(s) relevant. Maps provide the possibility of gaining an understanding of a complex problem situation

and, at the same time, facilitate a common understanding of specific issues among different stakeholders.

Concepts maps (Novak 1998), for example, are used to represent and organize knowledge of a particular topic. In concept maps, concepts are represented hierarchically, with the most general and inclusive one on the top and the more specific, less general ones underneath. Daley (2004) suggests that concept maps can be used to represent qualitative data at different levels of analysis. This has been taken up by Slaughter et al. (2006) to map different types of software development processes. The hierarchical ordering principle is, however, a serious limitation of conceptual maps when complex phenomena are to be mapped.

This is, to some extent, also valid for causal maps (Armstrong 2005) where concepts are ordered in networks from right to left to indicate causality. While this might be helpful for straightforward problems, for complex phenomena the search for causality might limit the analysis.

Recently Pries-Heje (2004) has introduced a knowledge mapping technique for IT projects. He uses matrixes to match knowledge needs with knowledge sources. The technique distinguishes knowledge needs in the areas of business, technology and project work, but concentrates solemnly on people as knowledge sources. A representation of the complex organizational relationship of individuals, groups, organizational procedures, IT systems, artifacts, reports, etc., in knowledge flows, however, requires a more holistic approach to both the collection of data as input for the maps and its representation in the maps. It also has to allow for generalization based on specific issue expressed by different stakeholders, as well as for interpretation and negotiation.

The technique to mapping knowledge flows presented in this section has its origin in systems thinking (Checkland 1981; Checkland and Scholes 1990) and its well-known technique for visualizing and understanding complex problem situations called *rich picture technique*. A rich picture is defined as "the expression of a problem situation compiled by an investigator, often by examining elements of structure, elements of process, and the situation climate" (Checkland and Scholes, 1990). A rich picture seeks to outline a holistic presentation of a problem situation.

The rich picture technique requires a thorough data collection, for example, based on interviews with representatives of all involved stakeholder groups. A rich picture contains different viewpoints, potential disagreements or conflicts allowing for multiple perspectives at one time. The technique allows for both insiders and outsiders to draw a complex human activity system in a picture. The technique does not favor one way of actually drawing over another, but leaves this to the picture creators. However, the basic elements characterizing the situation have to be visualized for the picture to serve as a means of communication. Even if all those involved do not understand the picture in the same way as the creators or as it was intended, the drawing itself serves as an enabler for a discussion,

which may support the development of a common understanding of the problem situation.

Our mapping technique is also inspired by the earlier work of Lanzara and Mathiassen (1985) on mapping problematic situations within software development projects. They distinguish diagnostic, ecological, virtual and historical maps. A diagnostic map consists of a root-cause analysis in which the mapmakers discuss experienced problems and seek causes and effects to find alternative approaches to avoid problems. Ecological maps outline the connections between problems and the organizational context of the problems. Virtual maps outline desirable future situations. Historical maps have a retrospective perspective as they map the past; a previous situation is described with respect to its key events to learn what might be critical factors in a similar future situation.

An important aspect with regard to map making is to acknowledge the map-making process itself as an important part of the result. The maps are the tangible results, but those involved in the creation process learn about each other's viewpoints and beliefs, and the process thus facilitates the exchange of opinions, the sharing of knowledge, and the development of a mutual understanding.

To analyze the knowledge flows in an organization, we utilize the strengths from each of these techniques. The resulting map of the knowledge flows and its constituting elements we call a map of knowledge flow or, in short, a knowledge map. The term knowledge map is widely used in different contexts. When we searched the term 'knowledge map' on Google in October 2004 we had 24,800 hits returned. Examples of knowledge maps include tools to support mind mapping, category-based indexing, and alternative methods of text representation. However, it also fits our purposes.

The rich picture technique provides the ability to visualize the different knowledge elements in a single drawing, while mapping techniques provide different analysis approaches for the actual drawing process and the drawing itself. A knowledge map thus consists of the elements from a rich picture: elements of structure, elements of process, and a representation of the climate within which these two exist. Whereas the actual choice of drawing symbols is not important, it is important to choose representations that are understandable, and direct the viewer's attention towards the relevant elements of the experienced practice.

The structural elements of a knowledge map constitute the basic nodes of the map. They consist of the different actors and groups involved in the organization which comprise the formal organizational constructs like the organizational units, project teams, individuals, etc. Important artifacts regarding the flows also have to be considered, for example, reports or software tools like an error reporting system. Knowledge 'storage bins' (Walsh and Ungson 1991) are also good candidates for structural elements in the knowledge maps. According to Walsh and Ungson (1991), knowledge is stored in the following six bins: (1) in the organization's individuals, i.e., in their belief structures, their memory stores, and in their personal records and files, (2) in the organizational culture, i.e., in the

way employees are taught by the organization to perceive, think, and feel, (3) in the transformations in the organization, i.e., in the actual transformation processes carried out in the organization, e.g., in standard operating procedures, (4) in the organizational structure, i.e., in the roles and rules constituting the organization, (5) in the ecology, i.e., in the physical layout of the organization, and (6) in so called external archives, e.g., governmental regulations, and former employees.

Wenger's (1998) description of communities of practice points to another important component of knowledge flows. He emphasizes that it is the belonging to several communities of practice that constitutes people's worldview and thereby their capabilities to understand the environment they see. Wenger refers to the communities as containers of competencies. The competencies evolve inside the communities, but are also exchanged at the boundaries between various communities. According to Wenger, the communities of practice, the boundaries between them, and the communities' members' identities play an important part in the social learning systems that constitute, among other things, an organization. This suggests that any representation of knowledge and knowledge flows is incomplete without depicting communities of practices and the interactions within and between them. Thus, groupings of employees, formal and informal, and their roles, formal and informal, constitute important elements of such descriptions together with the flows of knowledge that exist within, as well as between, these groups.

The central process elements of the knowledge map are the knowledge flows, i.e., the communication and the exchange of information, etc., which flow directly and indirectly between the structural elements. The flows constitute, in the terms of Huber (1991), the basis for knowledge acquisition, information distribution, information interpretation and organizational memory. A flow comprises of interaction between various structural elements, and can consist of informal discussions, as well as of, for example, strictly formal half-year reports; what is important is that some actor acknowledges it as a means of knowledge exchange. Some flows are bi-directional, and some unidirectional, and some might be both, depending on who defines them. The flows can differ with respect to their frequency and the amount of information they contain, both important features to provide an understanding of the overall flow between various elements. It is useful to model the types of knowledge contained in a flow. The importance ascribed to flows by different stakeholders is also a significant feature as it can bring potential misalignments into the open.

Knowledge flows are also related to learning and learning cycles. Learning occurs in the interplay between competence and experience (Wenger 2000), and often leads to a change in the repertoire of future behavior (Walsh and Ungson 1991). This interplay is, because of its cyclical nature, also referred to as the learning cycle (Hedberg 1981). Learning takes place and knowledge is gained when experience affects the cause-effect relations in the memory, which again changes the repertoire of competencies for future actions which—when applied—leads to new experiences, etc. The learning cycle constitutes the

basis for approaches to organizational learning. Wenger (2000) uses the term 'social learning' for this organizational learning cycle and describes how social learning occurs in the interplay between personal experiences and social competencies. A prerequisite for learning is that the learning cycles are complete, which implies that their various elements must somehow be connected, see also chapter 4. Experiences must be available to compare or learn from the effects of actions. Competencies must be available for deciding or learning which actions might be useful in a certain context. On the organizational level, this means that both experiences and competencies must be exchanged and shared to permit learning to take place.

The situation's climate is a key information provider. It contains expressions about the circumstances under which knowledge flows take place. This contextual information is a major indicator for pointing out problems. It comprises multiple perspectives depending on the viewpoints brought forward. It can consist of thoughts about why a situation is experienced as good or bad, thoughts of how a certain situation could be improved, expressions of where conflicts arise, or other comments about the knowledge flows. The techniques for constructing diagnostic maps and ecological maps are important tools in this part of mapmaking. By describing the climate, it is possible to gain insight into potential strengths or weaknesses of the knowledge transfer and sharing, which are important parts of the learning cycles in organizations.

There are many different ways of applying the technique; section 4 describes the concrete way that we applied it. However, it always incorporates participant involvement as a necessary condition to strengthen the relevance and validity of the map. The therapeutic effect of discussing and cooperating in map making is an important side effect of the technique.

During or after the map making the map can be analyzed with respect to potential problems, as well as the equally important sections of well-performing knowledge flows. Incomplete learning cycles might indicate improvement areas, whereas some parts of the organization might provide good examples for other parts.

4 The Pilot Test: Mapping knowledge flows in practice

The mapping was developed for SSE. The company was founded in 1985 and develops complex IT solutions within information and communications systems. It employs about 300 people, mostly system engineers with an academic background. All development is organized in projects and the project teams are quite stable: most developers only work on one project at a time. The projects are large, 55% of them larger than 10,000 working hours. Continuously, 20-25 customer projects exist simultaneously; a typical project develops new applications for one customer, but some of the applications are off-the-shelf products with long lasting projects for maintenance and further development.

The company is highly involved in developing its abilities to produce software; 10 years ago it adopted the capability maturity model (CMM) (Humphrey 1989) as a basis for conducting software process improvement efforts (Aaen et al. 2002), see also chapter 11. At the time of the intervention the company was at CMM level 3, but it later reached level 5. The organization has followed knowledge management principles for a number of years and has a fairly developed knowledge management system (Arent et al. 2002). The efforts are rooted in a SPI team with about 15 full-time staff.

The development application of the knowledge mapping technique was triggered by a situation where the members of the company's SPI team had experienced occasions where knowledge about development practices and improvement proposals did not flow as desired between development projects and the SPI team.

First, a thorough data collection was conducted to examine if and to what extent these issues were shared with other parts of the organization, and if so, how others understood and explained them. Based on this broad conception of the situation, the knowledge mapping technique as described in the previous section was developed and then applied as a means to investigate the organization's knowledge sharing capabilities and to identify possible areas of improvement.

The actual approach to applying the knowledge mapping technique consisted of two phases, a preparation phase conducted by a map maker, and a collective mapping session where two leading members of the SPI team and four members of the research team created the map in a one-day meeting.

4.1 Preparing the Map

The preparation phase was carried out by one of the authors who had also conducted the data collection. He created a preliminary knowledge map of the organization (figure 1) based on the data he had gathered as a preparation for the joint mapping session described in the next subsection. The purpose of this map was to serve as a guideline to be followed during the joint mapping session and to keep the task on track even when discussions would move in different directions. Preparing the map in advance ensured that all aspects relevant for the original map maker were covered. It allowed the map maker to prepare a note of questions on topics which he felt were not explained satisfactory in the collected data. Drawing the map in advance also provided the opportunity to list what he saw as major problems and improvement areas. These were used as discussion topics, in situations where the process needed stimulation. The preparation of a map in advance, without interference from others, also enabled him to record his understanding, and thus provided the whole joint mapping session with the quality of an outsider look at the organization. This external input would not have been as clear if organizational actors had had the opportunity to influence his views in advance.

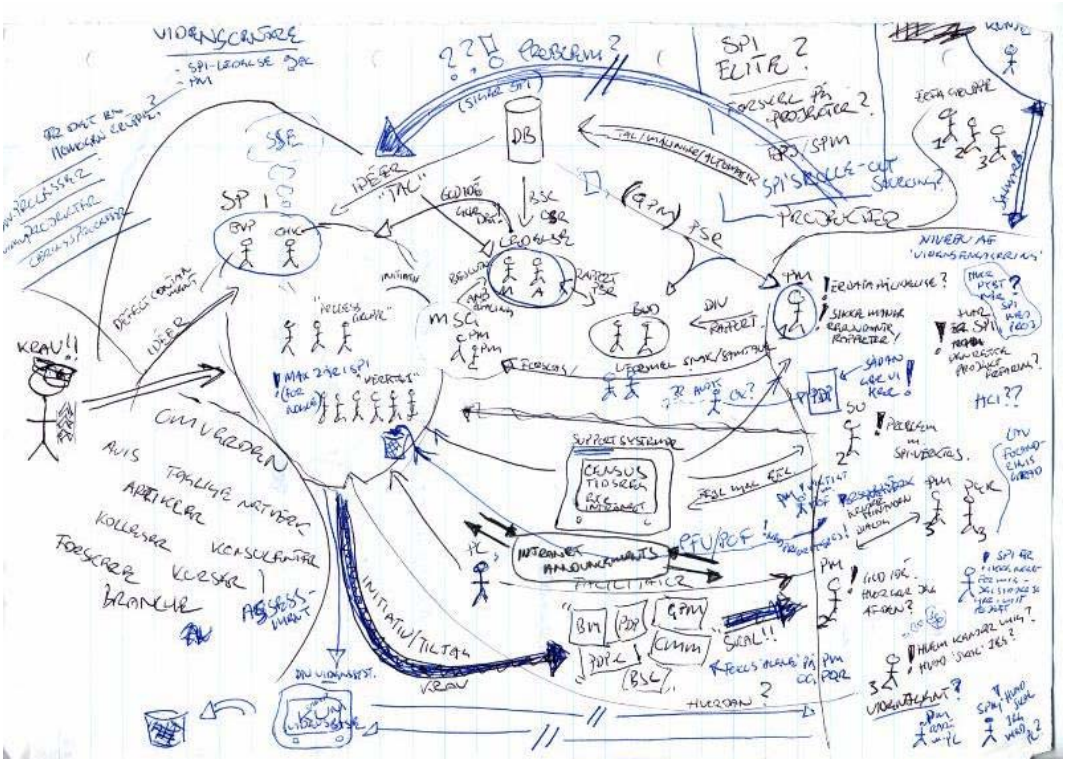

Figure 1: The preliminary map

4.2 Producing the Map

The second phase, the actual mapping session consisted of four steps and took place in a one-day meeting. The steps together functioned as verification, clarification, and extension of the map maker's preliminary map. The various elements of the map were discussed in an open atmosphere. The map maker acted as the meeting leader who also introduced and facilitated the process. Because not all stakeholder groups were represented, the map maker at the outset (in his own map) brought the individual viewpoints of these groups forward. He was also responsible for leading the discussion, and for documenting the results on a whiteboard.

The first step consisted of drawing all important elements of structure on the board. The mapmaker selected one area of the organization to begin with and started drawing organizational units, artifacts, people, etc., listed on the preliminary map. While doing this, he presented his understanding of the role of each of these. This promptly initiated a discussion among the participants because in their opinion some of the descriptions were not

correct. To represent individuals, we used 'stick figures,' some with additional characteristics because they represented individuals with specific roles or importance. We used groups of stick figures to represent certain organizational units, like development projects; sometimes we encircled these to mark specific boundaries. We used a picture of a document to represent written reports, and their formal abbreviations to distinguish them. Furthermore, we used various symbols for technical systems; these symbols were easy to understand for all participants, and were, if necessary, labeled with explanatory text.

The second step consisted of describing all different knowledge flows. The meeting leader introduced a specific flow between two or more people or artifacts, and described his understanding of it. This quickly facilitated a discussion outlining special cases and corrections, providing clarification and richness to the map. This step also introduced overlooked people, roles, and artifacts to the map.

The third step concentrated on the climate providing the context for the knowledge flows. For this purpose characteristic statements identified during the data analysis were added to the map. The step included a discussion of which flows were found problematic, and which flows were missing. When we conducted this step a lot of new problematic issues surfaced. The organizational actors strongly disagreed with the collected data and started defending their views. The meeting leader steered the discussion away from this position, and emphasized the opportunity to develop ideas for improvements. The focus was directed towards addressing the question of why some organizational actors experienced these problems, even if the SPI team's members did not acknowledge them.

Here, the mapping technique showed its value, since it was possible to demonstrate that some might experience problems or conflicts, whereas others did not see them. A new color was used to highlight the problematic areas and denoted with a large exclamation mark. Finally, this step was also used to indicate on the map where new ideas and initiatives originated by marking these with a light bulb.

The fourth step consisted of analyzing the identified problems in order to understand their roots and causes. This discussion supplemented the map with a list of identified improvement areas. The map allowed the diagnosis of each problem with its particular context with respect to structure and process, which made it easier to identify the parts of the organization that were affected, and should possibly be involved in the search for a solution.

Even though the steps above are described as a linear process, the actual mapping was characterized by allowing the discussion to follow interesting topics, and thereby mapping larger organizational 'chunks' rather than finalizing each step at a time. Thoughts were allowed to drift, and the discussion moved more iteratively from one topic to another. The meeting leader had his own map to fall back on when the process needed to pro-

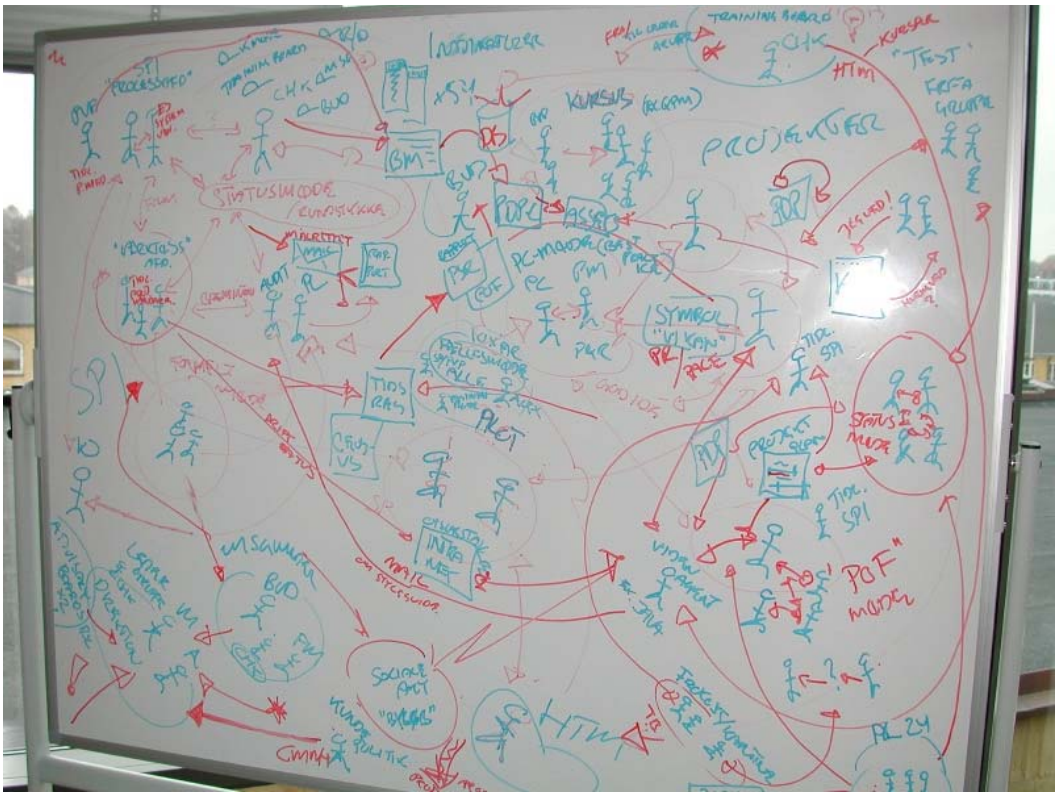

Figure 2: The produced map

ceed. To allow for further analysis and to serve as the documentation of the mapping process, the last item of the agenda consisted of photographing the map on the whiteboard.

4.3 The Results of the Mapping

A crucial result of the analysis was the finding that knowledge and key information regarding development projects were collected in formal project reports. The project managers responsible for creating the reports, however, felt that these reports never reached the places in the organization where they could be utilized and transformed into (new) organizational practices. As a consequence, the project managers spent less time providing this knowledge and information, thus making fewer experiences available. On the other hand, the readers of the reports felt that these offered very little of relevance, and therefore did not spend much time analyzing them. The recognition of this incomplete learning cycle led to the development and implementation of a new project evaluation concept, integrat-

ing the project reports and the diffusion of experiences through the establishment of formal knowledge networks (Hansen 2007).

We also recognized four categories of characteristic or critical situations, which even on our complex map were rather distinct:

- A specific individual or organizational unit with a large number of connecting knowledge flows, we call a *hub*. In our case, i.e., one person had several roles in various parts of the organization, which appeared in the map as many flows leading to him. A hub might be useful to have in an organization, if it can cope with the knowledge flowing to and from it, and can effectively use this information, but too many flows ending in one place might easily create congestion and thus a hub might end up developing into a bottle neck slowing (or discharging) information, and thus become a hindrance for the organization's ability to share knowledge

- *Black holes* are places where no flows origin. This means that knowledge only flows one way towards these areas. This might not be problematic, but, learning from experience is an important part of the organizational development, and when specific parts of the organization are not feeding back experiences, it is not possible for other parts to learn. An example from our study showed that one of the organization's knowledge sharing tools was not used by the employees making it a 'write only' asset from which no knowledge was fed back to the organization.

- Areas from where lots of flows originate, but none are oriented towards, are *springs*. These might indicate potential innovative centres where lots of ideas are created and exported to the rest of the organization. A spring might also point to an area, which is not using others' experiences. This is not necessarily a problematic situation, as it can represent a highly specialised unit which does not need any input, but it could also point to a potential problem. In our case, the top management constituted a spring by continuously feeding the organization with new ideas and suggesting new initiatives.

- *Missing links* describe situations where a link would be beneficial, but for some reason is not there or not functioning satisfactorily. As such, missing links are more problematic to spot in a map. Similar organizational units might achieve advantages from having a close dialogue and sharing of their experiences; if no knowledge flow exists between them, a potential benefit might be lost. In our study many projects found it difficult to feed back their experiences to the SPI team, which then had fewer experiences to base their work upon.

A final result of our analysis, which confirms the map's function as a device to support understanding and consensus, was the broader comprehension of the knowledge flows by and among the participants of the session. When the participants were presented with viewpoints other than their own, they had to reflect upon these, and this had an impact

upon how they saw the situation, and provided them with a better basis for an analysis in the future.

5 Discussion and Conclusion

Based on the insight that understanding the knowledge flows in an organization will support software development practices, we described in this chapter how in close collaboration with the employees of a software development company we identified possibilities for improvements. For this purpose, we introduced the concept of knowledge flows and developed and applied a knowledge mapping technique which visualized: the relevant organizational units, individuals and artifacts, the knowledge flows between them, and the climate and context in which these flows take place.

In this context we introduced the concepts of springs, hubs, black holes, and missing links which showed to be helpful to structure the discussion of knowledge flows. We used a range of qualitative data gathering methods throughout a 10 month period to inform the application of the mapping technique in the organization.

We conclude that this mapping technique provided a helpful means to understand the complexity of the knowledge flows in a development company with many simultaneous projects. The technique produced valuable results when it was applied in the company as it provided understanding and information about where and which improvements could be relevant.

In the described setting the knowledge mapping technique provided the organization with useful results and valuable feedback, but the same setting might not lead to the same achievements in another organization or with other participants. The technique as such is no silver bullet and no guarantee for success. Long-term cooperation with and the extensive presence of the researchers in the company, mutual respect, trust and a sincere atmosphere had developed and allowed for an open dialogue and discussion through data collection and, in particular, throughout the final mapping session. The researchers and especially the creator of the preparatory original map were acquainted with the organization's culture and overall values, its language and concepts, as well as their procedures and tools. This facilitated the mapping process, which was focused and progressed easily without significant problems. But the close relationship with the company and the employees also presented a danger. The apparent inside knowledge could lead to misconceptions and a distorted image of the organization. Here, though, the collective mapping session, in which matters could be openly discussed by representatives of different stakeholder groups, could function as remedial action.

However, our approach, which relies on and addresses all relevant personnel to get a view of the practice as perceived by all stakeholders, might—at least for the pilot experiment—be criticized as having only SPI management, that is, no ordinary project members were present at the mapping session. One might argue that involving more personnel in

the mapping session and providing more resources for the process could have avoided this. In our case, resources were limited; thus, we decided that for a pilot test of the technique a smaller forum would be adequate. The conflicting demands were balanced and compensated by the map maker, who, although he was an outsider, was well acquainted with and trusted in the organization. In this way an outcome was achieved that was not restricted by management views and limited input.

In this context, the use of drawings also has to be considered carefully. Checkland and Scholes (1990) underline that some people can easily handle representations such as rich pictures, whereas others have problems with drawing and discussing them. In our case the researchers all had experience with rich pictures as one form of graphical and visual aid and could thus help the other participants with the use of the technique. It was also beneficial to support the use of pictures with a meeting leader and let him draw the first version of the picture.

For SSE, the value of the technique was confirmed by the fact that the resulting improvement proposals were positively received and accepted by other members of the organization than those who participated in the mapping session.

More generally, our technique contributes to the existing body of knowledge in the field of knowledge-based software process improvement with a new promising support tool and, as such, could contribute to practical improvement work. To further develop and refine the technique, additional research testing and experimenting with it in different forms and within other contexts is, of course, necessary.

References

Aaen, I., M. Christiansen and H. C. Kristensen (2002). The Ambitious Effort: Stalemates and insider solutions. In: *Improving Software Organizations*: *From principles to practice,* L. Mathiassen, J. Pries-Heje and O. Ngwenyama, eds. Boston, Addisson-Wesley, pp. 65-82.

Arent, J., M. H. Pedersen and J. Nørbjerg (2002). Strategies for Organizational Learning in SPI. In: *Improving Software Organizations: From Principles to Practice.* L. Mathiassen, J. Pries-Heje and O. Ngwenyama, eds. Boston, Addisson-Wesley, pp. 235-253.

Armstrong, D. (2005). Causal Mapping: A discussion and demonstration. In: *Causal Mapping for Research in Information Technology*. V. Narayanan and D. Armstrong, eds. Hershey, PA, Idea Group Publishing, pp. 20-45.

Checkland, P. (1981). *Systems Thinking, Systems Practice*. West Sussex, UK, John Wiley & Sons.

Checkland, P. and J. Scholes (1990). *Soft Systems in Action*. West Sussex, UK, John Wiley & Sons.

Daley, B. (2004). Using Concept Maps in Qualitative Research. In: *Proceedings of the 1st International Conference on Concept Mapping.* A. Cañas, J. Novak, F. González, eds. Pamplona, Spain, 2004.

Hansen, B. and K. Kautz (2004). Knowledge Mapping: A technique for identifying knowledge flows in software organizations. In: *Software Process Improvement - Proceedings of EuroSPI 2004.* T. Dingsøyr, ed. Trondheim, Norway, Springer Lecture Notes in Computer Science, pp. 126-137.

Hansen, B. and K. Kautz (2005). Analysing Knowledge Flows as a Prerequisite to Improve Systems Development Practice. In: *Proceedings of the 13th European Conference on Information Systems,* Regensburg, Germany.

Hansen, B. (2009). Improving Software Process Improvement with Knowledge Management. *PhD Thesis.* Department of Informatics, Copenhagen Business School, Denmark (forthcoming).

Hedberg, B. L. T. (1981). How Organizations Learn and Unlearn. In: *Handbook of Organizational Design.* P. C. Nyström and W.H. Starbuck, eds. Oxford, Oxford University Press, 1: 3-27.

Huber, G. P. (1991). Organizational Learning: The contributing processes and the literatures. *Organization Science*, 2 (1): 88-115.

Humphrey, W. (1989). *Managing the Software Process*. Reading, MA, USA, Addison Wesley.

Lanzara, G. F. L. and L. Mathiassen (1985). Mapping Situations Within a System Development Project. *Information & Management*, 8(1): 3-20.

Mathiassen, L. (2002). Collaborative practice research. *Information Technology & People*, 15(4): 321-345.

Novak, J. (1998). *Learning, Creating and Using Knowledge: Concept Maps™ as facilitative tools in schools and corporations.* Mahwah, New Jersey, Lawrence Erlbaum Associates, 1998.

Pries-Heje, J. (2004). Managing Knowledge in IT Projects. *The IFCAI Journal of Knowledge Management*, II(4): 49-62.

Slaughter, S. A., L. Levine, B. Ramesh, J. Pries-Heje and R. Baskerville (2006). Aligning Software Processes with Strategy, *MIS Quarterly,* 30(4): 891-918.

Walsh, J. P. and G. R. Ungson (1991). Organizational Memory. *The Academy of Management Review,* 16 (1): 57-91.

Wenger, E. (1998). *Communities of Practice*. New York, USA, Cambridge University Press.

Wenger, E. (2000). Communities of Practice and Social Learning Systems. *Organization*, 7(2): 225-246.

7

Software Process Improvement Using Light Knowledge Tools

Jens Henrik Hosbond & Peter Axel Nielsen

1 Introduction

Since the mid 1990's several organizations have invested heavily in off-the-shelf knowledge management systems. The considerable interest in tool support for knowledge management in organizations has led to several reports on unsuccessful implementations of such software systems. Schultze and Boland (2000) report success rates as low as 30% caused by technologists' lack of ability to understand the work practices of knowledge workers. This has also been supported by others' research (Alavi and Leidner 1999; Swan et al. 1999).

In software engineering a common example is that of the experience factory (Basili et al. 1994; Basili and Green 1994; Rus and Lindvall 2002). The idea of the experience factory is to capture, explicate, and disseminate information and knowledge within the software organization. The experience factory's knowledge base is managed by a team of people in charge of collecting, storing and updating information in a database. Knowledge management is taken to be concerned with what should be stored in the database and how the data in the base should be searched (Althoff et al. 2000; Conradi and Dingsøyr 2000; Lindvall et al. 2001). There is, however, a scarcity of reports on successful implementations of the experience factory. Desouza (2003) provides three reasons for this, of which the most significant is that knowledge cannot be sufficiently captured and categorized in existing knowledge management systems and a literature review shows that most research on knowledge management in software process improvement conclude how difficult it is

to provide computer-support because knowledge management is very complex and person-dependent.

In software engineering and software process improvement, knowledge management is central and is important for the success of these activities (Mathiassen and Pourkomeylian 2003). Studies of software process improvement (Arent and Nørbjerg 2000; Baskerville and Pries-Heje 1999; Kautz and Thaysen 2001) call for a 'softer' approach to knowledge management than what has previously been seen in software engineering.

In the research presented in this chapter we pursued such a softer approach to knowledge management. Our aim was to explore and contribute to tool support for knowledge-based software process improvement in a way, which would contrast the experience factory. We conducted an action case study in one of Scandinavia's largest software development companies, henceforth referred to as ScanSoft, over a 1 year period. During this study we have focused on software process knowledge and improvement activities in ScanSoft.

The chapter is structured as follows. Section 2 presents the research approach of the study. A description of the case is provided in Section 3. Section 4 gives an analysis of the identified improvement issues, the design of the prototypes for tool-support as well as their evaluation. Section 5 follows with a discussion of the findings. The final section addresses our conclusions.

2 Research Approach

The research has followed an action case approach, a hybrid research approach mixing action research, i.e., intervention, and case study research, i.e., interpretation (Vidgen and Braa 1997). The study was conducted in 2002 and 2003, comprising three consecutive empirical activities: a case study, a soft systems analysis, and prototype experimentation.

To understand the knowledge processes within the SPI initiatives we conducted a case study of the current practices together with an analysis of the historical and cultural background of the organization. The empirical data were collected by means of semi-structured interviews with project managers, systems developers, and a chief information officer. All selected interviewees were selected had close relation to or were affected by the SPI initiatives in ScanSoft. An interview guide was designed and he themes in the guide included historical and cultural background, current work practices, knowledge creation and knowledge sharing, and SPI initiatives. Secondary data supplementing the interviews were collected from project-specific documents as well as from general organization-wide information. The data analysis was performed inductively. All interviews were transcribed and the loaded into the software tool HyperResearch. HyperResearch was used to organize the vast amount of interview data and to allow for automated coding and analysis of the data. The data analysis resulted in a set of concepts characterizing the software development culture in ScanSoft.

The data analysis was then used as point of departure for the action part in which we used the Soft Systems Methodology (SSM) (Checkland and Scholes 1990). Addressing the intangible nature of knowledge-related problems, the Soft Systems Methodology enabled us to consider several perspectives of the problem situation. This approach was beneficial as company culture, communication style, management style all influence how knowledge is created and shared. The modeling of systems in the Soft Systems Methodology lead to a comparison between the ideal expressed in the models and the actual state of affairs in ScanSoft. During this comparison we focused in particular on knowledge activities in need of improvement and possible tool support and developed ideas for knowledge tools.

These were then designed, realized and evaluated in 'light' prototypes. The experimental approach with the light prototypes followed the general ideas for prototyping in (Budde et al. 1992) and in particular we followed the activities suggested by (Mathiassen et al. 1998): planning, development, preparation, experimentation and evaluation.

By combining SSM and prototyping, we utilized both the analytical strength of SSM and the experimental strength of prototyping.

3 ScanSoft and It's SPI

We now introduce the case and summarize the initial findings of the case study, which built the foundation for the subsequent soft systems analysis and the prototyping of the knowledge tools.

3.1 Historical Background

ScanSoft is a Danish software development division within a large Scandinavian software company (for anonymity called ScanSoft in this chapter). The company was founded in the late 1960's. In the beginning of the 1970's it expanded into different regions of Sweden. During the latter part of the 1970's, the company became international by expanding to other Nordic countries, including Denmark, where it merged with ScanSoft, which then already was an experienced software house. By 1985, the company employed approximately 275 people. Through the 1980's and 1990's the company grew significantly. At the latest count in 2007 it employed around 7000 employees in Scandinavia and had a turnover of approximately €1 billion.

ScanSoft had attempted an ISO9000 certification in the early 1990's that failed. The failed attempt created a widespread resistance towards similar improvement efforts. The experiences with the certification attempt also created shared skepticism among top managers and project managers with respect to the organizational fit of the rigorous quality management procedures of ISO900 standards and maturity models like the Capability Maturity Model (Humphrey 1986). Consequently, software process improvement was

not prioritized and supported by ScanSoft's management. However, software process improvement was defined and developed by a few dedicated project managers acting on their own.

3.2 Software Process Improvement

The analysis of the case data resulted in the following concepts that characterize the software development culture in ScanSoft: team spirit, informal communication, personal networks, isolated projects, grass root software process improvement, and communities-of-practice. Of these concepts, 'grass root software process improvement' was particularly relevant for improvement efforts. The grass root initiative grew out of a dedicated group of project managers, who called themselves P+. This group had the inclination to improve the way experiences and knowledge about software processes were created and shared among the project teams. In particular they paid special attention to younger and inexperienced project managers with aim to improve this staff's competences.

Experienced project managers within P+ would from time to time have face-to-face meetings with more inexperienced project managers with the purpose to support, and tutor the inexperienced project managers through a dialogue. P+ members were therefore also regarded as mentors within ScanSoft.

Occasionally, members of P+ met to discuss many of the issues that they became aware of during these sessions. Based on these discussions smaller software process improvement initiatives were defined, planned and realized.

3.3 Knowledge Sharing in P+

The SPI initiatives of the P+ group were grass roots driven and they were not financially supported by nor were there any commitment form top management. Hence, we regarded the SPI initiatives as being informal in nature and not organizationally anchored. Instead, they were targeted at addressing and improving issues found at the project level.

As mentioned above, P+ was a group of highly experienced project managers, who had a long history in the company. P+ was established specifically to address the sharing of experiences between projects and to act as support for inexperienced project managers. The basic idea was that a group of experienced project managers would act as collaborative partners for project managers around the organization. Concerns and issues could be raised with P+ and experience and knowledge could be shared through the group. P+ consisted of 6 project managers. The P+ members used only a small part of their time in P+. The rest of the time they managed their own projects. It was deliberately decided that the P+ members should not dedicate a significant part of their time to support other project managers and run the risk of becoming a staff function or a quality management function.

Thus, the two main goals of P+ were: facilitating and sharing of experiences between projects and acting as support for inexperienced project managers. These goals were addressed through four activities:

1. Meetings where initiatives were defined, planned, committed to and later evaluated
2. Sparring sessions between experienced project managers and inexperienced project managers
3. Kick-off seminars for new projects
4. Thematic seminars on best-practices and experiences.

4 Design and Evaluation of Light Knowledge Tools

With a particular focus on P+ and following SSM, we created a so-called rich picture illustrating the different activities in P+ and the activities related to software process improvement. The activities included theme seminars, sparring sessions, P+ meetings, and startup seminars. Each of these were characterized by one or more of the five elements of knowledge management—capturing, sharing, creating, acquiring, and using knowledge as proposed by Kautz and Thaysen (2001).

From this rich picture, six system ideas were selected by P+. Of the proposed six systems, two were then selected for prototype experimentation. The selection process was inspired by Checkland and Scholes's (1990) model for comparing system ideas and it was conducted as a meeting with P+. P+ members commented on the system ideas and, based on their comments, an agenda system and a war stories system were selected for prototyping.

4.1 The Agenda System

The agenda system was proposed for supporting the preparation phase of P+ meetings by facilitating discussion and coordination of topics for the coming meeting. In effect, the intention was to make way for a more effective meeting process, based upon mutual interest and preparation among meeting participants. The agenda system was formally defined by the following definition of a human activity system :

> A human activity system to manage the creation of group agendas, by systematizing and prioritizing agenda items that may be discussed and commented upon by participants, in order to organize meetings and foster commitment and the creation of a shared context for all participants.

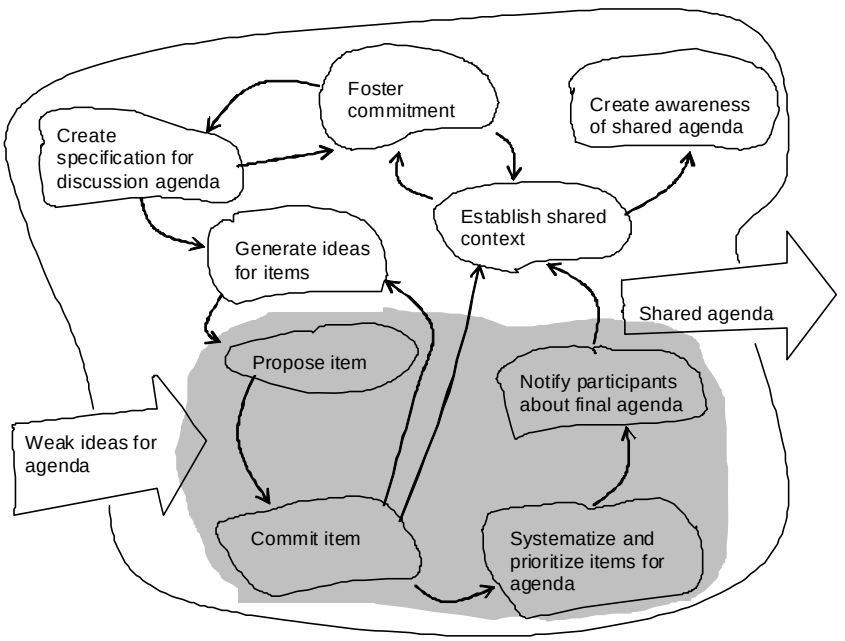

Figure 1: The human activity system for creating a shared agenda

To augment the understanding of the idea behind the human activity system and to illustrate the potential of supporting P+ meetings again using SSM a system model was developed. The system model is depicted in figure 1.

The 4 activities depicted as a grey area in the figure were identified by the P+ project managers as the activities which should be supported by the knowledge tool.

4.2 The Agenda Tool

The prototype of the agenda tool was developed as a web-based application following a client-server architecture. The client-server architecture was chosen to ensure instant corporate-wide use and easy maintainability of the system. The prototype was designed to resemble the look-and-feel of a standard Windows file explorer (see figure 2); that is, an overview of prior and future agendas was listed on the left, whereas the agenda topics associated with each agenda was shown on the right. Items on an agenda wee structured as a hierarchical tree of threads, following the common style of mainstream discussion boards. The tree structure was used as an invitation for discussion of items on the agenda for upcoming P+ meetings.

P+ meetings were not held on a regular basis and the use of the tool was therefore bound to be limited. Hence, it was a reuirement that the prototype was easy to use and

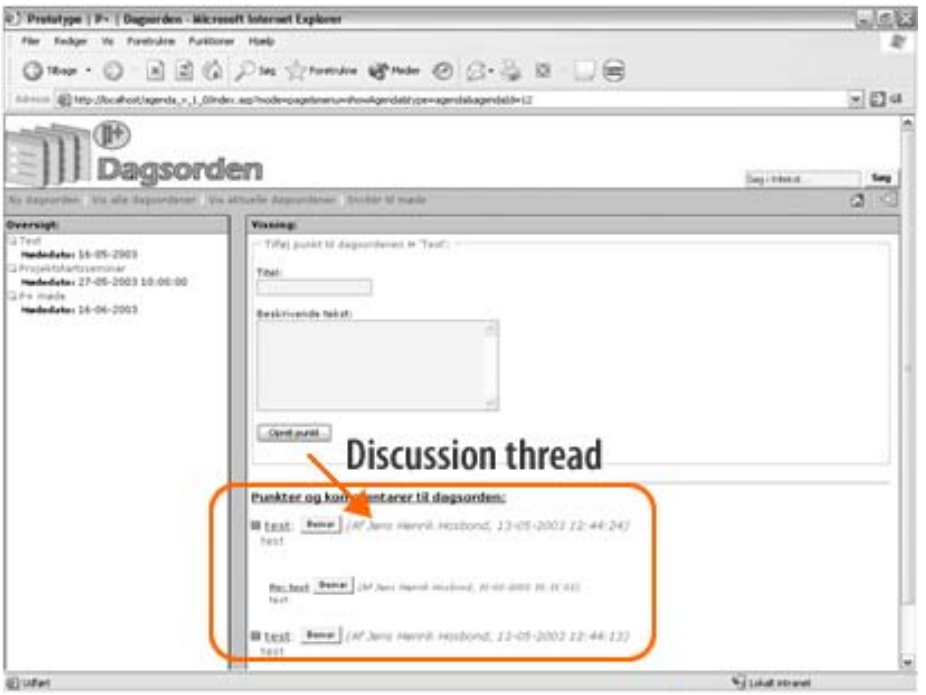

Figure 2: Agenda prototype: An example of a discussion thread

that no tool training was necessary. As the grey area in figure 1 show, the prototype was supposed to support the creation of an overall agenda for a P+ meeting. The creator of an agenda is also responsible for planning that specific agenda, but not for all other agendas. The planner may then invite P+ members to the discussion of the agenda. Invited P+ members are then able to comment on existing proposed agenda items and may also suggest new items for the meeting. Lastly, having discussed the content on the agenda through the agenda tool, the meeting planner decides the final items for the coming meeting. The sequence of items may be re-organized before accepting the final structure and content of the agenda. With the final agenda in hand, e-mail invitations can be sent out, with the agenda attached and with expected participants as receivers. .

Evaluting the developed, the general consensus among P+ members on the relevance of the agenda tool was that the prototype provided interesting support to the problematic situation of generating group agendas especially when colleagues were not co-located. As a P+ member stated:

> "It was pleasing to look at and very easy to grasp. I think we can easily use this. It could actually be very interesting."

Despite its apparent simplicity, the prototype appeared to help establishing a shared context prior to P+ meetings. Effectively, this would cater for a more effective meeting process. This was emphasized by another P+ member:

> "By using the agenda tool [prototype], people would be more focused when meetings start. Furthermore, the participants would have a shared understanding of what the topics are for the meeting and what is going to be addressed."

All in all, the prototype was well received and the tool deemed both relevant and useful.

4.3 The War Stories System

The idea behind the war stories system was to facilitate the transfer of individual tacit knowledge into an explicit form. A key enabler for this knowledge process is story telling. Story telling and particularly the telling of the emotional form of stories, also called war stories is widely used in communities-of-practices (Brown and Duguid 1991, Mathiassen 1998). ScanSoft was no exception here and with a strong autonomous project culture such war stories reflected experiences that had made a strong personal impression on the storyteller. In a project culture, as in ScanSoft, war stories convey both good and bad experiences.

War stories were highly relevant for the inexperienced project managers, but they may also be used as a common point of reference in general discussions of project management strategies and tactics. The war stories were supposed to mediate a meaningful and rich structure for explicitly communicating experiences, i.e., tacit knowledge from development projects, without losing too much situational context. Based on this the war stories system was defined as the following human activity system:

> A human activity system that organizes work-related experiences, by means of codifying, sharing and evaluating war stories, in order to obtain a common understanding of best and worst practices and experiences.

Similar to the agenda system, a system model presenting the activities of turning tacit into explicit knowledge was developed. It is depicted in figure 3. The grey area shows again the activities, which should be supported by the knowledge tool.

4.4 The War Stories Tool

The design style of the prototype followed the layout of the agenda prototype. The primary reason for doing so was that it created an obvious synergy between the two prototypes. The project managers should feel confident using both prototypes so the look-and-feel of the prototype largely resembled that of the agenda prototype. The war stories prototype was also built using a web-based client-server architecture.

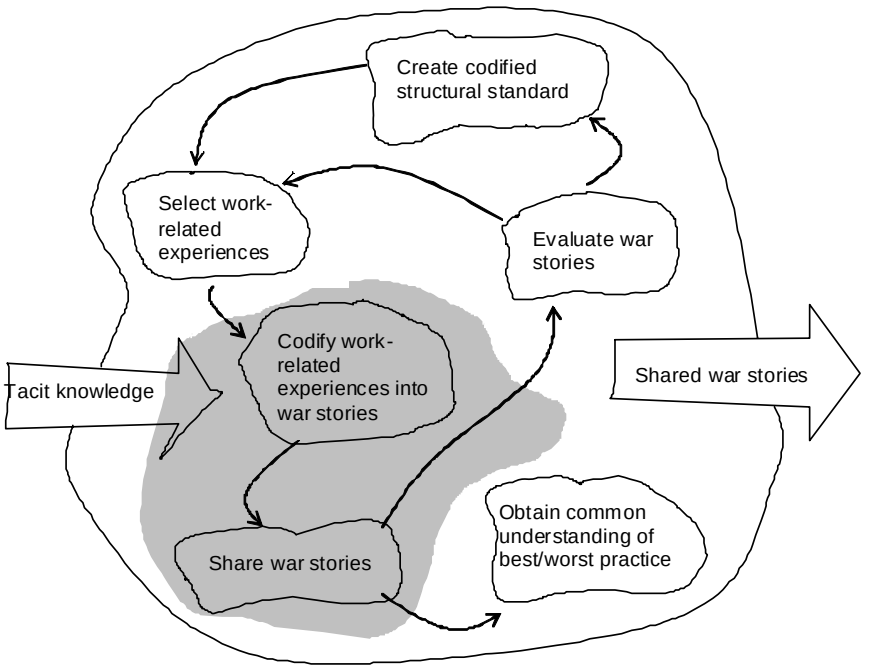

Figure 3: Conceptual model of the war stories system

The original war stories prototype had a very fluid narrative structure for the explication of war stories. There were no restrictions as to how the stories were told and structured. This, however, introduced the problem of interpretation. How should each story be interpreted and understood? By keeping a completely flexible structure, the context of the war story can indeed be very hard for the reader to grasp. Interpretations would be very diverse depending on the individual reader. A first evaluation therefore concluded that wa common structure of the stories is therefore necessary both for establishing a common foundation for explicating (author) and also for interpreting stories (reader). The relevance of the war stories prototype and the concern regarding the structure of war stories was commented on by a P+ member:

> "The war stories very much resemble what we are doing in daily practice. The goal must be to communicate a structural format of the stories that is adapted to the organization as a whole. The extraction of knowledge is important to the individual, but the knowledge extracted must be shared with others, thus-the presentation is important."

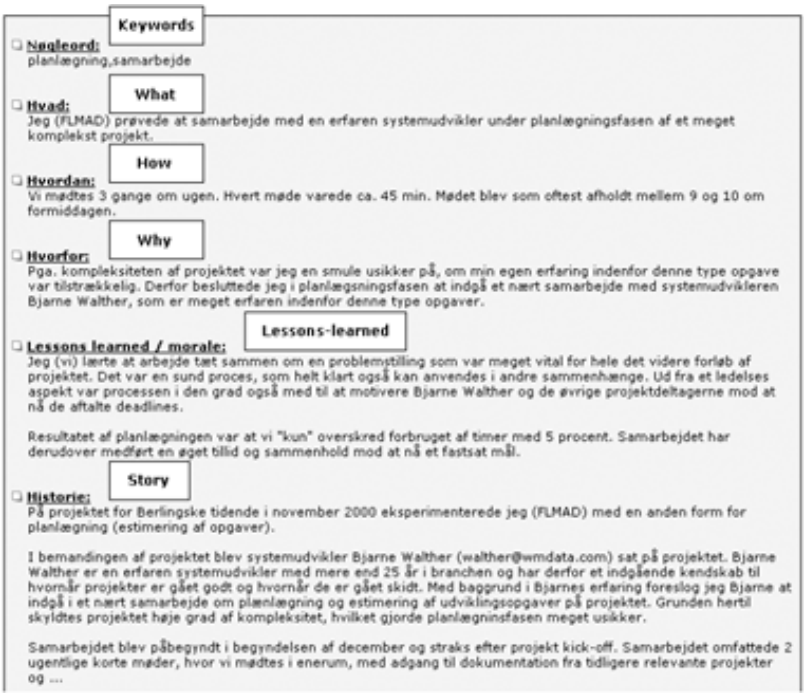

Figure 4: War stories prototype: Example of codification structure

After the evaluation of the first war stories prototype, a formal structure was added to the war stories. A story now contained six mandatory elements: keywords, what, how, why, lessons-learned, and story (see figure 4).

The idea was that to add a common structure for each war story, would creatw a familiar format, and thus enable a common way of interpreting the stories. However, the feedback from the evaluation of the second prototype was mixed. A primary concern was the possible semantic ambiguity in each of the six structural elements. For instance, does the 'how' relate to how the incident was experienced or to how the incident was solved? An attempt to solve the ambiguity problem was made, by adding popup windows, explaining the meaning of each element. However, as a P+ member noticed about the second prototype:

> "The use of 'what', 'how', and 'why' must be tried out in practice. If people write non-comparable descriptions in these fields, either the semantics of the fields has to be more clearly defined or they can be completely removed. "

This illustrates that even the basic requirements for the war stories tool were hard to define. The second prototype left the designers with quite some uncertainties regarding the

structural format of the stories and thus with the scheme for supporting the codifying of the project managers' experience into war stories. A third prototyping iteration could unfortunately not be made due to the closure of the research project. More structures should have been evaluated in the search of a format fitting the specific needs for capturing the essentials of good and bad practice in each software development project; Perhaps the whole human activity system for war stories should have been re-thought. The human activity system was built on the assumption that it was straightforward for project managers to explicate their experience, which is otherwise more or less tacit. The knowledge work involved in formulating the war stories takes skill and effort. In this respect the second prototype was still quite simple and seemed to not to provide adequate support.

5 Discussion

For the evaluation of (the usefulness of) the knowledge tools we relied on the statements from the involved project managers. In the following discussion we will reflect on the knowledge tools based on these evaluations and the knowledge processes in ScanSoft.

The differences between the two knowledge tools and their underlying human activity systems are interesting and can be summarised as in table 1.

The project managers in P+ evaluated the agenda tool quite positively. We attribute this to four elements, which are necessary, but none of which is sufficient for the adoption and utilization of the tools. The look-and-feel of the agenda tool was similar to their existing tools and thus was deemed as a feature that would reduce the learning effort. In the case of ScanSoft this happened to be the feel-and-look of a MicroSoft Windows user interface. The agenda tool is to be used to manipulate an agenda that gradually converges into the final agenda for a meeting. The project managers already knew the structure of an agenda and they knew how to work with an incomplete agenda. This meant that the new knowledge tool easily could support the knowledge process of agreeing on an agenda for a meeting. That the project managers knew the structure of agendas was also meant that no discussion what an agenda looks like was necessary and that the requirements were fixed before the first prototype was developed. The evaluation confirmed that this feature was stable and certain. It also lead to the judgement that the tool was or would be effective in the sense that its use would yield the expected results (i.e., an agenda), which could be used. It was also assessed as efficacious in the sense that it represented a proper means for the project managers in the knowledge process. There were also some minor doubts whether with an established agenda planning process the tool in its current form would be efficient on the long term, Nevertheless, in summary, we believe that the positive evaluation of the rather simple agenda tool can be attributed to the fact that the knowledge processes were known and were perceived as simple by the project managers.

	Agenda tool	*War stories tool*
Knowledge processes	Known and simple to project managers	Unknown and complex to project managers
Features evaluated positive or certain	• Windows look-and-feel • Resemblance with existing knowledge processes • Structure of agendas • Effective and efficacious	• Windows look-and-feel
Features evaluated negative or uncertain	• Long-term efficiency	• Resemblance to existing knowledge processes • Structure of war stories • Effective, efficacious, and efficiency

Table 1: Summary of differences between the two knowledge tools

The project managers in P+ evaluated the war stories tool as interesting, but had several reservations. The reservations concerned first of all the structure of the war stories. Steps have been taken in what appeared to be a useful direction. In the end, there was a suggestion for a codification scheme, about which the project managers agreed that it contained relevant formalizations of the context for the war story itself. It was however doubtful whether this would be sufficient to support the authoring project managers to structure their narratives as This may of course become easier when the war stories tool gets populated with exemplary stories which may inspireprospective authors.. Based on the case study it seemed evident that a main reason for this was that the project managers had not yet established the knowledge processes that the tool was supposed to support. Hence, it was also uncertain whether the tool would be effective, efficacious and efficient.

For the war stories tool it would take several more iterations of experimentation before the requirements for the tool might become more certain. So far in the experimentation the project managers had expressed that the approach taken had been useful, as they had gained insights that they would otherwise not have had. In particular, they felt that the experiments with prototypes had been valuable. They had long wanted to develop a knowledge tool for war stories, but through the focused experimentation they now realised that it would take some innovative thinking and designing to arrive at a useful tool for this.

Both the prototyped knowledge tools were 'light' in the sense that they were easy to develop and hence also easy to modify. In particular the overall design idea had been to develop several light tools in parallel. The approach taken with the light tools contrasts the

traditional theory of organization-wide support through a large, integrated, monolithic knowledge management system, e.g., the experience factory as suggested by Basili et al. (1994) where large amounts of data are collected about software projects and it firstlater is determined figured out how to process and interpret the data.

We ensured that the designed tools were tailored specifically to support and improve a particular problematic knowledge process. As the involved project managers selected the knowledge processes for tool support, the issues of user acceptance and user commitment became of lesser concern. Our approach to design small, simple, and light tools could be implemented with less effort. The two prototypes built in this study had implementation times as low as one to two days for each. The user interfaces for the prototypes were reused to decrease the implementation time, but also to make it easier for the involved actors to master both tools. Finally, the small development effort allowed for extended experimentation, especially where it is was necessary as in the case of the war stories where the knowledge processes were not yet well-understood.

6 Conclusion

Our study in ScanSoft resulted in two experimental prototypes. One prototype was considered immediately useful and relevant by the involved staff members. The other prototype was considered interesting, but not yet sufficiently useful. We have discussed several reasons for this difference and how the knowledge tools relate to the knowledge processes. Our overall findings can be summarized as: (1) the study and the prototype experimentation revealed that if the knowledge processes already exist and are well-understood by the actors it is relative easy to develop a useful knowledge tool; (2) if, on the other hand, the knowledge processes are not well-understood or do not yet exist, then the development of even a light knowledge tool will take some time and iterations and might ultimately not lead to any useful outcome; (3) the experimentation with prototypes may reveal necessary features that cannot be surfaced by a traditional systems analysis; and finally (4) these light knowledge tools contrast the mainstream ideas of knowledge management in software process improvement.

References

Alavi, M. and D. E. Leidner (1999). Knowledge management systems: issues, challenges, and benefits. *Communications of the AIS*, 1(2es).

Althoff, K.-D., F. Bomarius and C. Tautz (2000). Knowledge Management for Building Learning Software Organizations. *Information Systems Frontiers,* 2(3/4): 349-367.

Arent, J. and J. Nørbjerg (2000). Software process improvement as organizational knowledge creation -a multiple case analysis-. In: *Proceedings of the 33rd Hawaii International Conference on System Sciences*. Wailea, Hawaii.

Basili, V., G. Caldiera and H. D. Rombach (1994). The Experience Factory. In: *Encyclopedia of Software Engineering* - 2 Volume Set, John Wiley & Sons, Inc.

Basili, V. and S. Green (1994). Software Process Evolution at the SEL. *IEEE Software*, 11(4): 58-66.

Baskerville, R. and J. Pries-Heje (1999). Knowledge Capability and Maturity in Software management. *The DATABASE for Advances in Information Systems*, 30(2).

Brown, J. S. and P. Duguid (1991). Organizational Learning and Communities-of-Practice. *Organization Science*, 2(1): 40-57.

Budde, R., K. Kautz, K. Kuhlenkamp and H. Züllighoven (1992). What is prototyping? *Information, Technology and People*, 6(2): 89-95.

Checkland, P. B. and J. Scholes (1990). *Soft Systems Methodology in Action*, John Wiley & Sons.

Conradi, R. and T. Dingsøyr (2000). Software Experience Bases: A Consolidated Evaluation and Status Report. In: *Proceedings of the Second International Conference on Product Focused Software Process Improvement*, LNCS.

Desouza, K. C. (2003). Barriers to effective use of knowledge management systems in software engineering. *Communications of the ACM*, 46(1): 99-101.

Humphrey, W. S. (198). *Managing the Software Process*. Addison-Wesley, Reading.

Kautz, K. and K. Thaysen (2001). Knowledge, learning, and IT support in a small software company. *Journal of Knowledge Management*, 5(4): 349-357.

Lindvall, M., M. Frey, P. Costa and R. Tesoriero (2001). Lessons Learned about Structuring and Describing Experience for Three Experience Bases. In: *Learning Software Organizations (LSO)*. K.-D. Althoff, R. L. Feldmann and W. Müller, eds, Springer Lecture Notes in Computer Science 2176.

Mathiassen, L. (1998). Reflective Systems Development. *Scandinavian Journal of Information Systems*, 10(1&2).

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen and J. Stage (1998). *Object-oriented analysis and design*. Aalborg, Denmark, Marko.

Mathiassen, L. and P. Pourkomeylian (2003). Managing knowledge in a software organization. *Journal of Knowledge Management*, 7(2): 63-80.

Nonaka, I. and N. Konno (1998). The Concept of 'Ba': Building a Foundation for Knowledge Creation. *California Management Review*, 40(3).

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

Schultze, U. and R. Boland (2000). Knowledge management technology and the reproduction of knowledge work practices. *Journal of Strategic Information Systems*, 9: 193-212.

Swan, J., S. Newell, H. Scarbrough and D. Hislop (1999). Knowledge management and innovation: networks and networking. *Journal of Knowledge Management*, 3(4): 262-275.

Vidgen, R. and K. Braa (1997). Balancing Interpretation and Intervention in Information Systems Research: The Action Case Approach In: *Information Systems and Qualitative Research* A. S. Lee, Liebenau, J., and DeGross, J. I., ed, Chapmann & Hall.

8

Social Networks Analysis in Software Process Improvement

Peter Axel Nielsen and Gitte Tjørnehøj

1 Introduction

Software process improvement (SPI) has long been a concern for software companies and for research. The development of the Capability Maturity Model, CMM, (Humphrey 1989; Humphrey 1992; Humphrey 2002) by the Software Engineering Institute sparked a huge interest in the field. The CMM family of models has also been supplemented with the IDEAL approach (McFeeley 1996) for how to utilise the CMM.

The focus in this paper is on SPI in small and medium-sized companies. A core characteristic of small software companies seems to be that they often face changing environments and are more vulnerable than large companies. Ward, addressing software development in small companies, suggests, "the processes by which software is developed are likely to change with circumstances—perhaps even change dramatically—even while general principles like the need for good communication remain constant" (Ward et al. 2001).

Within SPI there has long been the concern that the maturity-model approaches, like CMM (Humphrey 1989), are not sufficiently adequate for improving small software companies. An early survey shows major concerns that the CMM does not fit small software companies (Brodman and Johnson 1994). Several studies of SPI for small companies reveal their difficulties: small companies cannot necessarily afford the investment in SPI (Kautz and Larsen 1997); they lack SPI knowledge (Cater-Steel 2001); they see SPI as bu-

reaucratic (Kelly and Culleton 1999); and traditional SPI methods are too costly for small companies (Villalon et al. 2002).

The difficulties of small software companies cannot be attributed to the CMM-based approaches as there are reported examples of successful SPI, some which are CMM-based, and some which are not. Kautz et al. describe a successful improvement effort of a small software company, in which CMM was used for the initial maturity assessment and IDEAL was used to structure the effort (Kautz and Thaysen 2001; Kautz et al. 2001). Also Kelly and Culleton (Kelly and Culleton, 1999) report on attempts to develop and test new SPI approaches for small software companies relying on CMM. For example, a particular company in a case study chose an approach informed by the CMM, but based on modified principles: (1) maximise involvement, minimise disruption; (2) stress quality, not CMM compliance; (3) emphasise the advisory role; and (4) promote efficiency. In yet another study it is suggested that while CMM is used for assessment, it is necessary to supplement it with what the authors call an 'action package concept' to overcome small companies' lack of follow-through into action planning and action plan implementation (Villalon et al. 2002).

There are reports of successful SPI where CMM or similar maturity models were not used. Kautz has studied process improvement in three small companies (Kautz 1998; Kautz 1999). The success of SPI in these companies is attributed to four factors (Kautz, 1998): a tailored approach; an experience network between companies; external assistance; and partial external funding. In another study a medium-sized company's problems with current development processes were assessed with a technique for problem diagnosis, which was not based on a maturity model and many of the identified problems could later be alleviated (Iversen et al.1999; Nielsen et al. 2002). In this study the success is attributed to the particular way experience and knowledge was shared during the problem diagnosis.

Sharing knowledge, also sometimes referred to as sharing experience, is fundamental in all these reports. Hence, we have undertaken research to understand knowledge sharing better and in greater detail. In particular, here we report on an investigation on how knowledge sharing takes place in particular social networks in software companies and how social network analysis can be used to this end.

The concepts knowledge sharing and social network analysis are presented in more detail in section 2. In section 3, we outline our research approach and describe the data collection and data analysis. In section 4, we present the case of SPI in SmallSoft and how we used social network analysis these. In section 5 we discuss the role of social networks in SPI in general and how social network analysis was useful in SmallSoft in particular. The chapter ends in section 6 where the chapter is concluded.

2 Knowledge Sharing and Social Network Analysis

It appears that part of the success of SPI in small companies has to do with how knowledge is shared amongst the developers and managers partaking in the SPI effort. Knowledge management is a relevant perspective to apply in SPI in general. Kautz and Thaysen (Kautz and Thaysen 2001) concur with this and put forward that knowledge in SPI is not only to be seen as a simple commodity, but needs to be understood in a much broader and social context. Support for the idea that understanding knowledge management is a key to SPI can also be found in several other studies (Baskerville and Pries-Heje 1999; Pries-Heje and Pourkomeylian 2004; Mathiassen and Pourkomeylian 2003).

There are several explanations of why this is so. First, it has been established that software development depends hugely on communities-of-practice, which differ from the formal organisation (Mathiassen 1998 p. 88). Communities-of-practice create the specific context as well as the shared experience and understanding of their members in such a way that they shape how new or modified processes are adapted, implemented or rejected. Second, SPI is a problem-solving activity (Mathiassen et al. 2002, p. 4) where problems in software processes have to be identified, needs have to be understood, possible improvements have to be devised and prioritised, and actions to improve must be taken. All these activities require communication of different perceptions and interests, of plans and priorities, and of outcomes. Third, SPI is also a knowledge creating activity (Mathiassen et al. 2002, p. 7) where SPI knowledge needs to be elicited from experience, some experience has to be explicated, concerns for capture and quality of available knowledge have to be addressed, and validated feedback has to be provided. Fourth, organisational influence processes are important in SPI (Nielsen and Ngwenyama 2002). This study of influence processes concludes that it is crucial to understand the networks through which power and influence is exercised; but also that a major source of power is knowledge and communication skills.

We find it interesting to analyse the networks through which knowledge is shared and communicated in greater detail because we expect that it can advance SPI in general, and in small companies in particular. Social network analysis is a framework for such detailed analyses.

Social network analysis is a general framework and a set of techniques applied to study the relationships between organisational actors and their exchange of resources; see (Cross and Parker 2004; Wasserman and Faust, 1994). In social network analysis organisations are viewed as consisting of actors linked together in networks through action, exchange, and interpretation and sharing of resources like information and knowledge. Social network analysis seeks to provide a way to look at an informal organisation, which exists in parallel to the formal and hierarchical organisation chart. In this view, organisations are made up of interdependent actors with relational ties between them. Network models conceptualise structure as lasting patterns of such relational ties (Wasserman and Faust

1994). Wasserman and Faust further define actors as discrete individual, corporate or collective social units, (i.e., not only as a single person). The relational ties can be of varying types: evaluation of one person by another (as with friendship), transfer of material resources, affiliation, authority (as between managers and subordinates), and behavioural interaction like sending messages and engaging in a discussion (Wasserman and Faust, 1994, p. 18).

Social network analysis is not a new approach. It has been developed and applied in a large number of organisational studies (see Scott 2000; Tichy et al. 1979; Wasserman and Faust, 1994), but it has not been applied in SPI efforts before. The framework does not provide a unit of analysis and data may be collected about many different kinds of actors and relational ties. It is, however, common to collect data about the contents of the relational ties as well as their intensity and reciprocity. Based on the collected data, the approach requires the study of network properties and structural characteristics. For example, (Tichy et al. 1979) define some of properties as density, centrality, and star. These are just a few of the analyses that can be performed on the total network. The analyses all have a foundation in graph theory (Borgatti and Everett, 1992; Scott, 2000; Wasserman and Faust, 1994), but the interpretation and the semantic implication of these analyses remain specific to the setting where the data were collected.

Our application of social network analysis focuses on the social networks through which software process improvement may happen and in particular we focus on communication about SPI.

3 Research Approach

The research approach followed the ideas combined in an action case study (Vidgen and Braa 1997; Braa and Vidgen 1999). An action case study is partly action research and partly a case study. It is action research in the sense of (Checkland 1991; McKay and Marshall, 2001; Iversen et al. 2004), where interventions take place and scholarly knowledge is gained, but with a much shorter time-span and with lesser involvement of the researchers in the action, in our case the SPI activity. It is also a case study in the sense of (Yin, 1994; Walsham, 1995), that research beyond interventions takes place, but with a lesser depth in understanding the specifics of the case. The benefits lie in how case study and action research supplement each other.

The data collection and data analysis for the action case study was performed in two parts. For the purpose of understanding the case and the context for action the researchers collected data about the company, its SPI activity, the its history with SPI, and in particular all minutes from meetings in the SPI group, the progress reports, and other similar sources. The data were analysed informally to inform the researchers and to write the case report, which is summarised in section 4.

For the purpose of taking action informed by social network analysis the researchers collected and analysed the data following a more stringent procedure. The procedure is similar to that outlined by (Cross and Parker 2004 p. 143), and it contains the following steps:

1. Identify the group
2. Collect data about relationships
3. Visually analyse the results
4. Feedback the results to the group and validate the results
5. Evaluate the outcome

As we wanted to investigate to what extent and in which ways the company communicated and shared knowledge about software process improvement we identified the relevant group as all developers and all managers in SmallSoft.

The accompanying instruction told the respondent to also register the initials of their communication partners and use a new line for every interaction. The instrument provides this pattern for all interactions. This means that every reported line in a returned questionnaire is evidence for a relationship.

All questionnaire data were transferred to a tool that offers various display features and analyses, which are performed automatically by the built-in graph algorithms. The tool was used to analyse and keep an overview of the data (e.g., to select parts of the graph, show different attributes and weights, identify central actor, cut-point, etc.) using graphical elements to visualise structures in the social networks. The tool was also used to elucidate the several network structures like centrality, peaks, blocks, components, k-cores, etc.

The analysis of the network data was iterative. The researchers were consistently looking for patterns in the network models, which confirmed or rejected working hypotheses about the company's SPI activities. That led to analytical insight, which in turn led the researchers to modified and new working hypotheses. The iterative analysis was temporarily stopped to validate the findings with two department managers. Their feedback was used to extend the iterative analysis. It also gave a detailed impression of which network models were relevant from a management viewpoint as some of the models provided by the researchers presented an interesting hypothesis and a proper finding, but were not providing valuable managerial insight. The management's feedback also led the researchers to prompt several developers to respond to the questionnaire to increase the data validity.

The analysis ended with a second validation session was with all three company managers (for a description of the case company see subsection 4.1).

4 Social Networks in SPI

The modelling of social networks followed the approach outlined in section 3. In this section we first provide a case description as background for the social network analysis, then we present the models and analyses, and finally we report from the validation of the findings.

4.1 Case Background

SmallSoft is a small software company with two departments. The ERP Department develops a large ERP system and maintains it at a number of customer sites. The tasks are characterised by long-term and close contacts with a few large customers. The software developers have much domain knowledge within logistics in the particular area where their customers operate. The head of this department is also responsible for the quality system and the company's ISO9000 certificate. He was also heading the SPI group. The Tailor-Made Department develops several tailored systems for many different customers. Their products range from traditional administrative systems to web portals. The application domains vary and the developers' primary expertise lies within software engineering and project management.

Previously, improvements in software development were casual and spread through collaboration and informal contacts between colleagues. A few significant improvements had attracted management's attention and were turned into company-wide improvements. One company-wide improvement led to an internal software development project, which produced a support tool for tracking development tasks. Most improvements, however, were small and remained personal or local between a few colleagues.

When the research began, the company was introduced to a basic SPI approach and soon top management announced the slogan "CMM level three—in three years." A SPI group was formed and a developer from each of the departments was appointed to the group. The group took on the responsibility of assessing the current practices, planning improvement initiatives and implementing these. Successful improvements were supposed to be added to the existing quality system. The manager of the ERP department later characterised this new set-up as a complete failure. His perception was that some developers felt pushed aside and that others stopped focusing on improvements waiting for the results from the SPI group. The SPI group on their part lacked time and resources and organised only one improvement initiative. At the same time the company experienced a market decline and following low sales figures, this led to a shift of focus away from improvement activities and towards sales activities and monthly sales figures.

Despite these setbacks, SmallSoft's management recognised the value of their previous improvements as vital for their business success and found it necessary to proceed. The two department managers' shared perception was that future improvements had to

be rooted in a strategy that would provide faster feedback as well as visible and immediate benefits for the software developers. It was in this atmosphere that the analysis of social networks was initiated.

4.2 Social Network Analysis of SPI in SmallSoft

The analysis had the immediate purpose of understanding the specific company's social networks as a basis for managerial decisions about SPI. To that end we chose to visualize the models of communication networks that emerged from the data when displayed with NetDraw.

An initial analysis was performed after which the models, the analysis and the immediate findings were validated in a session with the SPI manager. The validation session led the researchers to another round of data collection to increase coverage and to make sure that most developers and managers had responded to the questionnaire.

The most basic network model is shown in figure 1. The node distribution feature in NetDraw provides its visual layout.

The model should be read in the following way. Circles and squares represent developers; circles depict that they are from the Tailor-Made department and squares show developers from the ERP department. Developers 29 and 30, depicted in black, are no longer with the company. Developers 4, 10-13 have not responded to the questionnaire and no others have reported on communication with these developers. Triangles are managers (9, 19, 21); manager 9 is the CEO. The number of respondents thus is 23 of 28 staff, or 82%. When these models were used in SmallSoft the real names of the staff members were shown.

The graph analyses follow (Scott 2000). The *initial graph* analysis was to look for components and central actors, because this provides a good overview to begin with.

The component analysis was based on the formal concepts of component, cut-point, and clique. A *component* in a social network is defined as a maximal connected subnet (Scott 2000 p. 101). The model of SmallSoft reveals that it consists of a single component because all developers and managers are related to at least one other except developers 4, 10-13 whom no one communicated with. These outliers as well as developers 29 and 30 were removed in subsequent analyses. A *cut-point* is a node whose removal would increase the number of components (Scott 2000, p. 107). In figure 2, manager 19 is a cut-point who would split the company in two components, and developer 5 is a cut-point who would disconnect developer 2 from the main component. Similarly developer 6 is a cut-point who would disconnect developer 14. A *clique* is a subnet in which every possible pair of nodes is directly connected and the clique is not contained in any other clique (Scott 2000 p. 114-115). Counting only those subnets with more than 3 nodes the following subnets are cliques: (16, 22, 24, 28); (15, 24, 25, 26); (5, 9, 19), (6, 7, 17, 18).

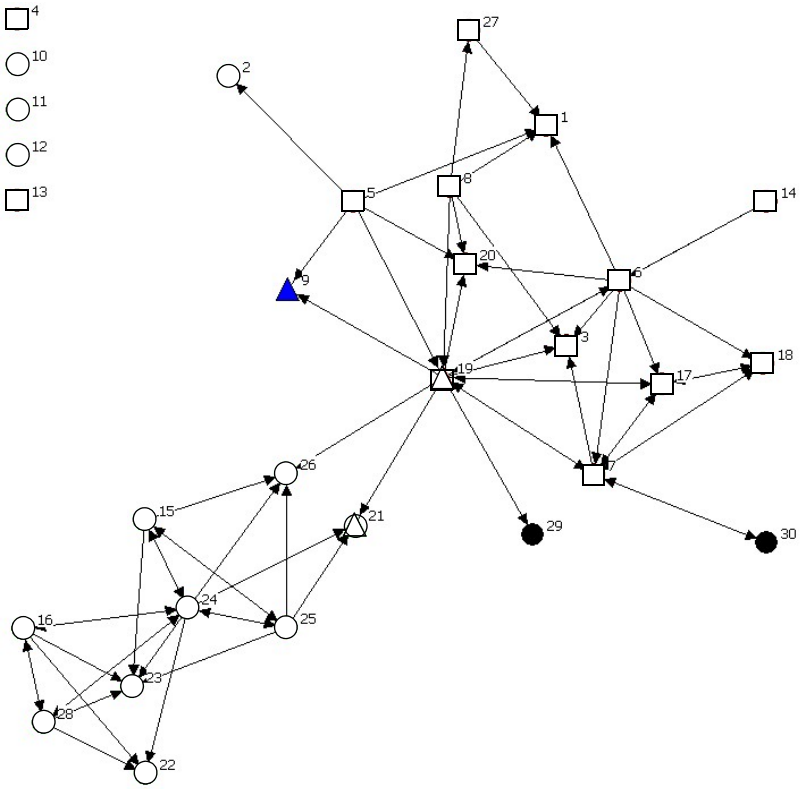

Figure 1: Basic communication network for communicating about SPI activity

The centrality analysis is based on the formal concepts of local centrality, global centrality, and peak. A node is central if it has a high degree, i.e., it has relations to many other nodes. Centrality based on degree can be measured locally or globally. The measure for local centrality is defined as the number of direct relations (Scott, 2000, p. 83). The nodes with the highest local centrality are: 19 (10); 24(8); 6(8); 7(6); 23(5). The measure for *global centrality* (or closeness) is defined as the sum of distance to all other nodes (Scott 2000, p. 86). The closer a node is to all other nodes, the more central it is; the closeness of nodes for all nodes in the SmallSoft network is as follows: 19(39); 26(44); 21(45); 6(48); nodes 5, 8, 3, 7, 24 have a a distance sum of below 53, while the rest are above. The global centrality can also be measured as betweenness of a node defined as the proportion of node pairs between which a node is (Scott, 2000, p. 87). NetDraw computed these to: 19(144); 24(60); 26(57); 21(41). A node is a peak if it is more central than any point it is (directly) related to (Scott 2000, p. 98). The peaks in SmallSoft are: manager 19 and developer 24.

Our qualitative analysis started with a working hypothesis that Smallsoft was a very informal organisation, however with strong monitoring by its department managers. Furthermore, prior to the social network analysis we held the perception that company management was in control and that all SPI activity had to be communicated through the managers. Consequently, we did not assume that there were social subnets with the capacity nor the inclination to communicate on SPI and to follow through to action on SPI.

Our subsequent analysis of the model led to the following results. As mentioned above, first of all it shows that five developers are completely outside all communication about SPI. Second, it identifies one main component containing both the ERP and Tailor-Made departments. Manager 19 is the central actor as he is the actor with the highest degree, i.e., numbers of ties: 10. Manager 19 is also a peak as he is more central than any other actor he is connected to. Developer 6, though highly connected with a degree of 8, is not a peak as he is connected directly to manager 19. This is not surprising as 19 is the manager of the ERP department and responsible for the quality system, the ISO9000 certificate, and also the SPI manager. He is connected to the top manager, CEO 9, and all connections between the ERP department and Tailor-Made department go through him.

Manager 21, the manager of the Tailor-Made department, is far from central and not a peak. He shares the contact with the ERP department with developer 26. In the Tailor-Made department, developer 24 with a degree of 8 is the only peak and he is connected to everyone in the department. The path from any of the managers to any of their developers is less than or equal to 2 edges. In the ERP department this is due to the central role of the manager and in the Tailor-Made department it is due to developer 24.

Manager 19 is the most important cut-point. If he is removed from the network, it will split into two components. Developers 5 and 6 are marginal cut-points as they will only cut out one other actor. Manager 19 scores very high on both closeness and betweenness. On closeness there is not really a significant difference between developers 26, manager 21 and developer 6. On betweenness manager 19 is undoubtedly most significantly with his score of 144; the important second person in this category is developer 24 with a score of 60.

4.3 Analysis of the Relation Attributes

Figures 2 shows the attributes of the communication for the main component. The differences between the models in figure 2 illustrates the differences between formal and informal communication and between written and oral communication. The communication is mostly informal and all actors are involved in informal communication. Formal communication is only found around the two peaks and between the two departments. Written communication has a stronger presence in the Tailor-Made department and around the manager of the ERP department. Oral communication is widespread and eve-

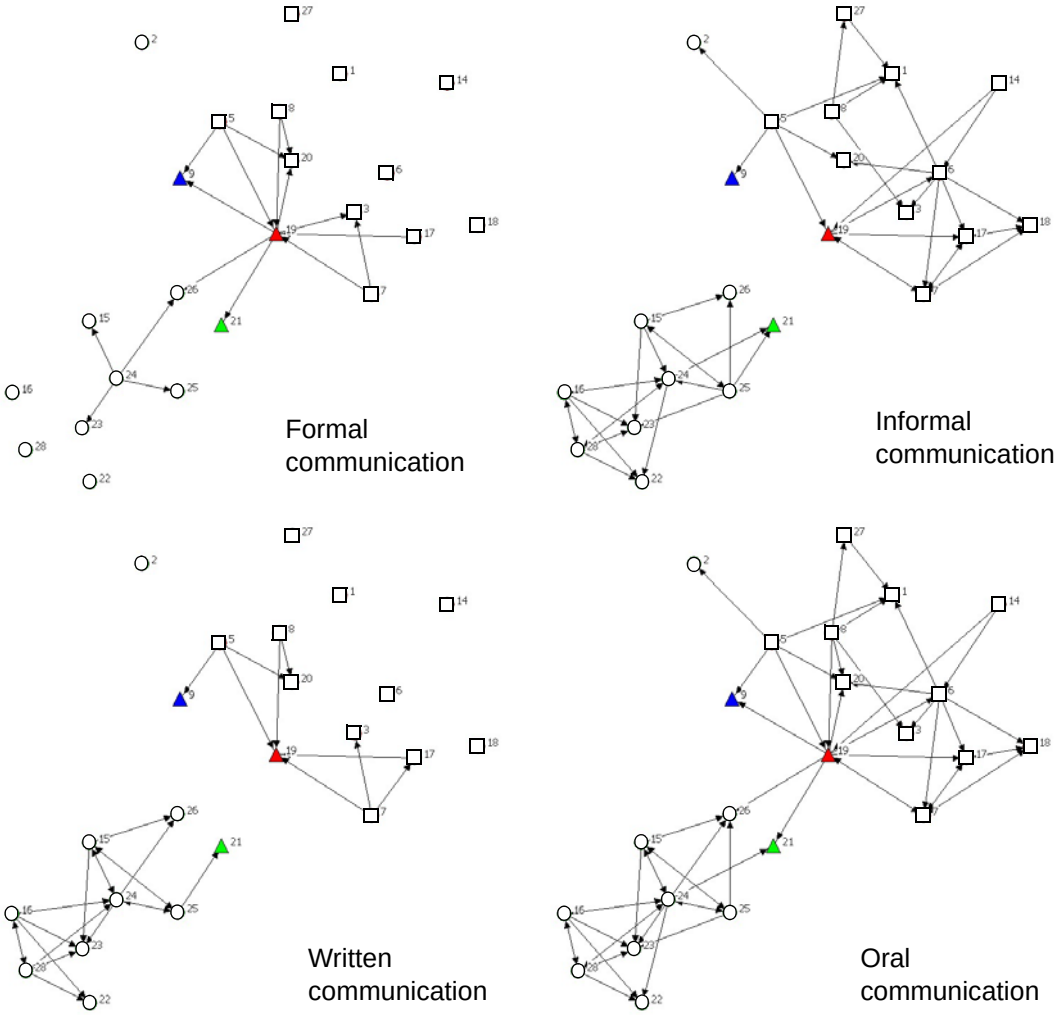

Figure 2: The models of attributes in the communications networks

ry actor participates in oral communication. It is worth noticing that the communication between the departments is formal but oral.

4.4 Summary of the Analysis

Overall the social networks show two departments with an informal, mostly oral and widespread interaction within the departments, but with sparse contact between departments and to top management. The ERP department has a central manager, 19, gatekeeping the department against all the other actors in the company in a more formal way

than usual in the company. He controls the communication on improvements both within his own department, but also at the management level and he is the only middle manager with contact to the top management.

The Tailor-Made department has a strong internal actor in developer 24 keeping the department together and communicating intensively with many other employees. The manager of the department, 21, plays a lesser role in SPI as he has fewer ties and partakes only in lightweight communication. He only connects to the whole department through developers 24 and 25. This looks like a widespread delegation of responsibility for SPI.

Until the time of the analysis SmallSoft had followed a centralised and formal improvement strategy. There are considerable misfits between a centralised strategy and the underlying social networks. This may largely explain the failure of the improvement effort so far. The underlying social networks are uneven and, in the ERP department, developers are unaccustomed to written communication while in both departments the networks are also lateral thus less disposed to acting on formal management directive. In contrast, the applied centralised SPI strategy is management-driven and communicated in writing and formal. The social network analysis thus leads to the conclusion that either the social networks must change or another strategy must be chosen.

Faced with these alternatives Smallsoft's management wants to change to a decentralised strategy. They feel that this will suit the company better and will involve more developers. When embarking on a decentralised SPI strategy they should keep in mind that:

- The remarkably weak ties between the two departments certainly hinder a central and cross-departmental SPI approach also in the future.
- A serious management commitment to SPI will be very difficult to exercise with so little communication on SPI involving the top manager; perhaps the lack of management involvement shows that SPI is not of strategic importance to the company's business strategy.
- Few improvements will spread easily from one department to the other; closer ties need to be built between the two departments and at the level of the developers; if this is impossible or undesirable, the departments should be seen as separate social networks and independent SPI activities should be organised in each department deliberately decreasing dependency on cross-department knowledge sharing.
- Any SPI initiative in SmallSoft will benefit from at stronger collaboration among the managers and also involving the CEO.
- The ERP department could benefit from decentralisation, less formalisation and delegation of responsibilities; manager 19 could very well be overloaded with responsibilities; if that is the case, he is a bottleneck that inhibits improvements and hinders knowledge sharing and communication in the department; management commit-

ment to SPI is based on real involvement and focus.

- The analyses do not display network structures that hinder ideas and improvements from being communicated among developers.

These advices for the SPI managers are very much in line with (Cross and Parker 2004). They suggest that it is a management task to initiate, develop and maintain networks. They further propose that the internal network structure of a company should be aligned with its environment. For a small company like SmallSoft the environment for the ERP department changes only slowly, but it is vulnerable to a few missed sales opportunities. In the Tailor-Made department there are often changes that it should respond. Management should hence consider whether they want to move developers between the departments.

5 Discussion

In the following discussion, we discuss first the findings from the action case in SmallSoft. Second, we focus on wider research implications and how the findings extend research on SPI in small companies.

The most significant finding of the social network analysis for SmallSoft was that there was already communication about SPI and that knowledge about software development was already shared. The managers already knew this in general, but they did not know the details. The network models showed many of these details which the managers were unaware of and which they would not address in their dealing with the software process improvement effort.

The social network analyses proved valuable in SmallSoft. They provided the researchers with substantial insight for their action research endeavour. They were also useful for Smallsoft's managers in several ways:

- The network models provided images of the communication about SPI which the managers trusted as they had been involved in their validation.
- The models contained angles, pointers and clues that the managers had never thought about before. The SPI manager in particular genuinely found the models interesting as a kind of mirror in which he could now see his own organisation in a new light.
- The models had been useful in communicating the findings from the researchers to the managers. They proved valuable as a starting point for the discussion of an appropriate strategy for SPI in the company, as they emphasized two major problems in the current situation that all managers could agree upon.
- The models were used as a basis for taking decisions about SPI and how to improve the underlying networks that had the potential of pushing the SPI effort forward.

> These decisions led to actions involving management more strongly and also to the creation of improvement teams across the departments.

It is evident from the SmallSoft case that communication and knowledge sharing in SPI is an integral part of SPI. This should be acknowledged by researchers and managers and more attention to should be paid to communication and knowledge sharing issues in SPI efforts. The research literature of small software companies has mostly been concerned with measuring maturity and the perceived problems r small companies have with in particular the CMM. There is research addressing the need for a closer look at knowledge management from a social perspective (see Kautz 1998; Kautz and Thaysen 2001); there is research addressing knowledge as a commodity to be stored in an experience base (see Conradi and Dingsoyr 2000; Rus and Lindvall 2002); there are reported experience from building knowledge networks in software organisations (see chapters 6 and 7). There are however no reports where the underlying communication networks have been analysed as we have done here.

Our study illustrates that modelling social networks is particularly relevant for understanding SPI activity in small companies. Small software companies are less likely to favour a formal, centralised SPI approach and SmallSoft is no exception here. It is thus reasonable to discuss how the lessons learned in our case study concerning communication, knowledge sharing and social network analysis may generalise.

Modelling social networks fits well with a low budget approach to SPI. It is cost-effective to analyse the underlying social networks that are an important part of the infrastructure for a more informal SPI approach in small companies, as these firms lack the economical inclination to invest in a formal, rational, centralised infrastructure.

Modelling social networks enables small companies to discuss and to exploit the possibilities that already exist and to focus on necessary improvements as a basis for SPI. We thus contention t that the way we have modelled social networks can be well transferred to other, similar organisations. Based on our experience, we suggest that it will work for small companies, but we can only speculate about whether it will also be feasible for large software organisations.

It is likely that the visualisations from the tool we used will be less useful with more than a hundred developers and other software packages for social network analysis might be more useful.

However, irrespective of the size of company our study shows that communication about SPI is also necessary in large organisations. What we do know so far is that in order to facilitate discussions that bring improvements forward, the network models must show the networks in a visual way that can be grasped by the involved actors without them being experts in social network analysis.

To understand the social networks and how they contribute to SPI can be a supplement to a CMM-driven strategy. In a CMM-driven strategy the focus is on processes and

much less on people independently of whether they are developers or managers (Aaen, 2003). A social network analysis offers the opportunity to focus simultaneously on how people communicate and how this communication supports knowledge sharing and as such becomes a prerequisite for the organisational change.

The data collection methods and the analysis, and use of models in discussions and reflections are not specific for SmallSoft. They are all transferable to other small software organisations. Thus, we claim generality for the applicability of modelling social networks and performing social network analyses. What cannot be transferred to other organisations are the specific models, the analyses of SmallSoft through modelling and the specific outcomes of the discussions.

6 Conclusion

In this chapter we reported an action case study in a small software company. Data were collected through a case study and a questionnaire that was designed specifically to get information about communication patterns in the social networks involving software developers and their managers. Our findings can be summarised as:

1. Communication about software process improvement follows other patterns than official and bureaucratic channels. It is important to understand the structure of these informal communication networks as they can promote or hinder a particular improvement effort.

2. Such communication networks can be studied through social network analysis. Social network analysis and its accompanying tools and techniques offer several very useful analyses. The managers in the case company appreciated these findings and consequently acted upon them. In particular they deliberately sought to remedy identified shortcomings in the network structures.

3. Social network analysis is very likely to be useful in other small organisations as data collection, visualisation, and the analysis techniques fit well with the particular challenges faced by small software companies, which want to engage in SPI.

The limitations of this action case study are related to its purpose, which has been to explore the usefulness of social network analysis for software process improvement. This exploratory study is only based on a single case and that limits its generalisation. In addition the original data collection was difficult as not all data was unambiguous due to the layout of the questionnaire However, our study shows a high validity due its high response rate and due to the fact that we co-operated with management who iteratively validated our findings during joint validation sessions. Although the analyses performed with the software tool are most reliable and it thus is likely that a repetition would reach the same out-

comes again, the results are only valid for SmallSoft. Hence we do not claim generality for them.

In a continued effort to make social network analysis more useful for software process improvement we will undertake further research with more software companies and improve the questionnaire as well as the particular analyses and their interpretations.

References

Aaen, I. (2003). Software Process Improvement: Blueprints versus Recipes. *IEEE Software*, 20(5): 86-93.

Baskerville, R. and J. Pries-Heje (1999). Knowledge Capability and Maturity in Software management. *The DATABASE for Advances in Information Systems*, 30(2).

Borgatti, S. P. and M. G. Everett (1992). Notions of Position in Social Network Analysis. *Sociological Methodology*, 22: 1-35.

Braa, K. and R. Vidgen (1999). Interpretation, Intervention and Reduction in the Organizatioanl Laboratory: A Framework for In-context Information Systems Research. *Accounting, Management and Information Technologies*, 9: 25-47.

Brodman, J. G. and D. L. Johnson, (1994) What Small Businesses and Small Organizations say about the CMM. Paper presented at the *16th International Conference on Software Engineering*, Sorrento, Italy.

Cater-Steel, A. P. (2001) Process Improvement in Four Small Software Companies. Paper presented at the *Proceedings 2001 Australian Software Engineering Conference*.

Checkland, P. (1991). From Framework Through Experience to Learning: The essential nature of action research. In *Information Systems Research: Contemporary Approaches and Emergent Traditions*, H.-E. Nissen, H. K. Klein, and R. A. Hirschheim, editors. North-Holland, Amsterdam, pp. 397-403.

Conradi, R. and T. Dingsoyr (2000). Software Experience Bases: A consolidated evaluation and status report. In *Product Focused Software Process Improvement* 1840, pp. 391-406.

Cross, R. and A. Parker (2004). *The Hidden Power of Social Networks: Understanding How Work Really Gets Done in Organizations*. Harward Business School Press, Boston.

Humphrey, W. S. (1992). Introduction to Software Process Improvement. Technical Report, CMU/SEI-92-TR-007, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA.

Humphrey, W. S. (2002). Three Process Perspectives: Organizations, teams, and people. *Annals of Software Engineering*, 14(1-4): 39-72.

Humphrey, W. (1989). *Managing the Software Process*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Iversen, J. H. L. Mathiassen, and P. A. Nielsen (2004). Managing Risk in Software Process Improvement: An action research approach. *MIS Quarterly*, 28(3): 395-433.

Iversen, J., P. A. Nielsen, and J. Nørbjerg (1999). Situated Assessment of Problems in Software Development. *The DATABASE for Advances in Information Systems*, 30(2): 66-81.

Kautz, K. H. (1999). Making sense of measurement for small organizations. *IEEE Software*, 16(2).

Kautz, K. (1998). Software process improvement in very small enterprises. *Journal of Software Process - Improvement and Practice*, 4(4): 209-226.

Kautz, K. and E. Å. Larsen (1997). Diffusion Theory and Practice: Disseminating quality management and software process improvement innovations. Paper presented at the *5th European Conference on Information Systems,* Cork, Ireland.

Kautz, K. and K. Thaysen, (2001). Knowledge, Learning and IT Support in a Small Software Company. *Journal of Knowledge Management*, 5(4): 349-357.

Kautz, K., H. Westergaard, and K. Thaysen (2001). Understanding and changing software organisations: An exploration of four perspectives on software process improvement. *Scandinavian Journal of Information Systems*, 13: 31-49.

Kelly, D. P. and B. Culleton (1999). Process improvement for small organizations. *Computer*, 32(10): 41-47.

Mathiassen, L., P. A. Nielsen, and J. Pries-Heje (2002). Learning SPI in Practice. In *Improving Software Organizations: From Principles to Practice*, L. Mathiassen, J. Pries-Heje, and O. Ngwenyama, editors. Addison-Wesley, Boston, pp. 3-21.

Mathiassen, L. (1998). Reflective Systems Development. *Scandinavian Journal of Information Systems*, 10(1&2): 67-117.

Mathiassen, L. and P. Pourkomeylian (2003). Managing knowledge in a software organization. *Journal of Knowledge Management*, 7(2): 63-80.

McFeeley, B. (1996). IDEAL: A User's Guide for Software Process Improvement. Technical Report, CMU/SEI-96-HB-001, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA.

McKay, J. and P. Marshall, (2001) The dual imperatives of action research. *Information Technology and People*, 14(1): 46-59.

Nielsen, P. A., J. H. Iversen, J. Johansen, and L. B. Nielsen (2002). The Adolescent Effort. In *Improving Software Organizations: From Principles to Practice*, L. Mathiassen, J. Pries-Heje, and O. Ngwenyama, editors. Addison-Wesley, Reading, MA.

Nielsen, P. A. and O. Ngwenyama (2002). Organizational Influence Processes in Software Process Improvement. Paper presented at the *European Conference on Information Systems*, Gdansk. S. Wrycza and K. Kautz, editors.

Pries-Heje, J. and P. Pourkomeylian (2004). Beyond Software Process Improvement: A case study on change and knowledge management. Paper presented at the *EuroSPI Conference*, Trondheim, Norway.

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

Scott, J. (2000). *Social Network Analysis: A handbook*. Sage Publications, London.

Tichy, N. M., M. L. Tushman and C. Fombrun (1979). Social Network Analysis for Organizations. *The Academy of Management Review*, 4(4): 507-519.

Vidgen, R. and K. Braa (1997). Balancing Interpretation and Intervention in Information Systems Research: The action case approach. Paper presented at the *IFIP TC8 WG 8.2 international conference on Information systems and qualitative research*, Philadelphia, PA.

Villalon, J., G. C. Agustin, T. S. F. Gilabert, A. D. Seco, L. G. Sanchez, and M. P. Cota (2002). Experiences in The Application of Software Process Improvement in SMEs. *Software Quality Journal*, 10(3): 261-273.

Walsham, G. (1995). Interpretive Case Studies in IS research: Nature and method. European *Journal of Information Systems*, 4: 74-81.

Ward, R. P., M. E. Fayad, and M. Laitinen (2001). Software Process Improvement in The Small. *Communications of the ACM*, 44(4): 105-107.

Wasserman, S. and K. Faust (1994). *Social Network Analysis: Methods and applications*. Cambridge University Press, Cambridge.

Yin, R. (1994). *Case study research: Design and Methods*. Sage Publishing, Beverly Hills, CA.

Part III

Tales

9

The Role of Improvisation in Adoption of Process Technology in a Small Software Firm

Gitte Tjørnehøj and Lars Mathiassen

1 Introduction

Software process improvement (SPI) has become an increasingly important technology for software firms struggling to stay competitive. The Capability Maturity Model (CMM) (Paulk et al. 1993) and the more recent CMM Integrated (CMMI Production Team 2000; 2002) have sparked a new discipline of engineering management that plays a dominating role in practice and research (Hansen et al. 2004). The CMMs are based on the ideal of a rational, control-centered culture for software development (Ngwenyama and Nielsen 2003), and although other SPI approaches have been suggested, they do not differ from the CMMs when it comes to underlying values (Hansen et al. 2004). While the literature offers a number of successful cases of adopting CMM (Carnegie Mellon 2005; Dion 1992; Humphrey et al. 1991; Wohlwend and Rosenbaum 1994), reports on failure and difficulties have lately increased (Hansen et al. 2004 Mathiassen et al. 2002; Rodenbach et al. 2000). Especially within small software firms, adoption of SPI technology seems problematic since these organizations lack sufficient resources to invest in improvements (Brodman and Johnson 1994; Brouse and Buys 1999; Steel 2004; Villalon et al. 2002), and SPI knowledge is not sufficiently tailored to their needs (Kilpi 1998; Saastamoinen and Tukiainen 2004). Small software firms are highly sensitive to dynamic environments (Ward et al. 2001), and dominating approaches to SPI fit poorly (Leung and

Yuen 2001; Varkoi et al. 1999) because it normally takes several complex and expensive initiatives to reach new maturity levels (Aaen et al. 2001). However, improvisation is often promoted as a means to resolve contradictions between environmental pressures towards innovation and lack of time, knowledge, and other resources (Chelariu et al. 2002; Ciborra 1999).

With this background, we have investigated the role of improvisation in SPI technology adoption over a ten year period from 1996 to 2005 in a Danish software firm, SmallSoft. The investigation is based on a longitudinal, interpretative case study (Pettigrew 1990) of SPI-related activities in SmallSoft. Focusing on key encounters that impacted engineering, management, and improvement practices within the firm (Cho et al. 2006; Newman and Robey 1992; Peterson 1998), we investigated the following research question: "Why, when, and how does improvisation shape the adoption of process technology in a small software firm?" The study helps us understand the role played by improvisation as small software firms struggle to adopt advanced process technologies in a situation where resources are few and knowledge not appropriate or readily available.

This chapter is structured as follows. First, we present relevant theory on SPI in small firms and on improvisation. Next, we present and explain the adopted framework for analysis of improvisation (Cunha et al. 1999; Kamoche et al. 2003). Based on our research approach, we introduce SmallSoft and provide a detailed account of how SPI technology was adopted over the period 1996-2005. Finally, we discuss the case in response to the research question and highlight the contributions and implications of the study.

2 Theoretical Background

Small software firms face special challenges when improving software practices. These challenges relate to the resources available and to having minimal influence over the environment. In the following, we review what is known about SPI in small software firms, and we present the improvisation framework that we have used as a lens to analyze the case.

2.1 Improvement in Small Software Firms

Managers in small software firms can find advice for SPI technology adoption in the literature. Most studies investigate small firms trying to adopt CMM or other rational software models. Typically, these studies describe the difficulties small software firms encounter and how these can be successfully resolved. Numerous studies suggest adaptations of CMM to fit small firms' needs (Batista and Figueiredo 2000; Casey and Richardson 2004; Horvat et al. 2000; Kautz 1999; Kautz et al. 2000; Kelly and Culleton 1999; Kilpi 1998; Varkoi et al. 1999; Wilkie et al. 2005). Paulk (1998) argues that small firms' adoption of CMM "may be different in degree, but they are not different in kind" from those of other

organizations. And, in general, it takes "professional judgment and understanding of how the CMM is structured to be used for different purposes" (Paulk 1998).

Another group of mainly European studies has discarded CMM and developed alternative approaches fitted to the market dominated by small and middle-sized companies, but still keeping the basic values of the rational software organization intact. Examples are DSDM and People Process (Coleman and Verbruggen 1998), TAPESTRY (Kuvaja et al. 1999) and SPICE (Truffley et al. 2004). This literature focuses on the downsizing of SPI technology to fit the resources of small firms, but do not address the culture differences, lack of knowledge, and external dynamic pressures that these firms experience. Moreover, most of the studies are based on observations over limited time periods focusing on initial adoption of SPI technology. Only a few studies cover a considerable time span and report on how SPI initiatives evolve over time (Balla et al. 2001; Kuvaja et al. 1999; Richardson 2001; Truffley et al. 2004). Thus, the literature provides little knowledge on how to achieve and sustain improvements over time in constantly changing environments. In this study, we have therefore given center stage to understanding how SPI technology was adopted in a small software firm over a period of ten years. To make sense of this adoption process, we focus on events in the firm's environment, and on why, when, and how improvisation played a role in shaping SPI practices and outcomes.

2.2 Organizational Improvisation

Improvisation has been pointed to as a way of coping with time pressures and rapid shifting environments combined with lack of resources, making rational planning, decision processes, and knowledge creation being difficult or even impossible (Ciborra 1999; Cunha et al. 1999). Improvisation is often seen as the deviation from the norm of planning and rational decision-making; but increased uncertainty, complexity, and environmental dynamics create new conditions for firms in which the ability to improvise becomes more important (Chelariu et al. 2002).

The concept of organizational improvisation is inspired by traditional Jazz (Barrett 1998; Mirvis 1998; Peplowski 1998). Later literature building on empirical evidence leads to a definition of improvisation focusing on convergence between planning and execution of actions (Crossan 1997; Crossan and Sorrenti 1997; Crossan 1998; Crossan et al. 1996; Miner et al. 1997; Moorman and Miner 1995; Moorman and Miner 1998; Orlikowski 1996; Orlikowski and Hofman 1997). Based on a comprehensive literature study, Cunha et al. (1999) suggest a definition of organizational improvisation as:

> ... The conception of action as it unfolds, by an organization and/or its members, drawing on available material, cognitive, affective and social resources (Cunha et al. 1999) .

The first part of the definition "The conception of action as it unfolds..." emphasizes that improvisation is deliberate, extemporaneous, and occurs during action. The second part "...drawing on available material, cognitive, affective and social resources" relates improvisation to bricolage by emphasizing that planning and action need to take place within the limits of available resources and knowledge. Improvisation both "departs from current routines/knowledge and builds on those routines/knowledge" (Cunha et al. 1999); thus, routines/knowledge potentially affects whether improvisation occurs and the quality and magnitude of it. Cunha et al. also discuss degrees, types, and measurement of improvisation and try to identify theoretically why, when, and how improvisation happens. They point to triggers and conditions for and characteristics of improvisation:

> ... thus improvisation arises when both (1) a demand for (a) speed and (b) action, and (2) an unexpected (and unplanned for) occurrence are perceived by the organization. (Cunha et al. 1999)

Weick (1993), however, emphasizes that improvisation can also be triggered purposefully. Unexpected occurrences triggering improvisations are not always perceived as problems, but could also be opportunities to take advantage of internal or external change.

An experimental culture, minimal structure, and a low procedural memory or a small number of routines are important conditions for the ability to improvise in organizations (Cunha et al. 1999). An experimental culture values action and experimentation when trying to understand and deal with reality. By minimal structure is meant a minimal control structure required for focusing, coordinating, and keeping the necessary feeling of urgency, still leaving room for participants to innovate. Milestones and goal setting are recommended as efficient tools. Procedural memory is the amount of routine knowledge that organizations possess. If procedural memory is low, it leaves more room for improvisation since more events are unplanned. On the other hand, a high procedural memory, perceived as adaptable knowledge instead of unbreakable rules, will also enhance improvisation.

Improvisation can have negative as well as positive outcomes in situations where it is necessary to create new knowledge. Positive outcomes include motivation, flexibility, increased ability to improvise, new knowledge about the world, as well as new routines and practices. However, organizations that engage in improvisation also face risks, including inappropriate learning biased by actual circumstances, opportunity traps by not acquiring new knowledge, over amplifying emergent events and addictiveness to improvisation thereby under-utilizing existing knowledge and skills; they also face increased anxiety and uncertainty for employees.

We have based our analysis on the conceptual framework originally presented by Cunha et al. (1999) and later utilized and presented in a slightly different form in Kamoche et al. (2003). Kamoche et al. (2003) added three different metaphors of improvisation (Indian music, music therapy, and role theory) to that of traditional Jazz, to provide a

spectrum of improvisation processes. We have drawn from both sources in the final analysis framework presented in table 1.

In this view, unexpected events can trigger improvisation when a need for immediate action is felt by organizational actors. Hence, the adopted framework starts by identifying such triggers. The following improvisation—if any—is then described in terms of actors, planning and execution of action, utilized material and resources, and finally products. Also, we characterize improvisation by degree, type conditions, and influencing factors, and eventually clarify the outcome of improvisation. This framework is appropriate in the context of this study since it is centred around events that spark action and, as such, fits well with our event-based analysis. The framework helps identify and characterize key improvisational elements in the investigated case of SPI technology adoption.

Trigger:	Description:	Characteristic:

• Unexpected event?
• Unplanned for?
• Need for action?
• Need for speed?
• Within action span?
	• What action?
	• Who acted?
	• How was planning and execution of action?
	• Which material and resources were utilized?
	• What were the products?
		• Degree
		• Type
		• Conditions
		  • Culture
		  • Minimal structure
		  • Procedural memory
		• Influencing factors

Positive outcome:	Negative outcome:	
• Increased flexibility • Learning • Emotional • Further motivation	• Biased learning • Opportunity traps • Over amplifying emergent events • Addictiveness of improvisation • Increased anxiety	

Table 1: Framework for analysis of improvisation adopted from (Cunha et al. 1999) and (Kamoche et al. 2003)

3 Research Approach

3.1 Longitudinal Interpretative Case Study

We have framed the investigation as a longitudinal, interpretative case study (Pettigrew 1990; Walsham 1993; Walsham 1995) based on a ten year-long collaboration with SmallSoft through a university-industry network (Mathiassen 2002). This approach allowed us to analyze in detail how adoption of SPI technology evolved over time and to investigate the role played by improvisation in that process (Cunha et al. 1999; Kamoche et al. 2003).

SmallSoft

SmallSoft is a small Danish IT firm with approximately 50 employees. Their core competence is to combine domain specific engineering knowledge with IT competencies to serve their customers by developing new solutions or by adapting standard software.

When a big industry-firm in 1987 went bankrupt, three of the employed engineers formed a new small consultancy firm (SmallSoft*)* operating in that same industry segment. They began to develop dedicated software tools in-house to support their consultancy, which soon led them to develop a material-management-tool as a joint venture with a main customer. Even though the virtues of this software product were its built-in domain knowledge, not its technical quality, it soon evolved into an important standard product, and the ad-hoc and improvising development process laid the basis for the software practices of today.

In the early nineties, software production was established as a separate business area and SmallSoft started to employ additional developers, and thus established a new department for tailored IT-systems. After a few years, the IT-business area dominated the firm that now had grown to 42 developers: 25 developers worked in the Standard Department responsible for the original and still important standard software that over the years had developed into a portfolio of subsystems and versions; 15 developers worked in the Tailored Department tailoring systems to a variety of customers; finally, a new department developed a storage management product based on a recent acquisition of a two person firm.

Under the impression of the bankruptcy the founder's formed a firm culture based on a low risk attitude and a belief in core engineering skills and long term personal customer relations during the first difficult years. Despite growth, SmallSoft's top management and board continued to exercise a defensive business and investment strategy, expecting a surplus every month. Employees were valued as innovative domain-knowledgeable experts, often working alone or in very small teams in close connection with customers. Software engineering skills were generally considered less important. During the mid nineties, the development of the standard product was fleshed out from development of customer versions to overcome severe quality problems. Within this subgroup of developers, software skills, software processes, and product quality became increasingly important.

Data Collection

Data were collected covering a period of ten years, where we periodically worked with SmallSoft as employees, university teachers, and researchers engaged in action research (Mathiassen 2002; McKay and Marshall 200; Susman and Evered 1978), described in greater detail below. Our relationship with the firm began in 1999 when the first author

was employed as project leader participating in implementing the new quality assurance system (QA system—launched in 1996) and ended in 2005 when an action research project was concluded. Through these activities, it was possible to collect a diverse and rich selection of data from a variety of sources covering a relatively long period of time. First, as part of a university-based action learning program, a group of employees at SmallSoft engaged in an SPI initiative starting in 2000, which was supervised by the second author; the resulting report included assessments of maturity, analysis of practice, descriptions of interventions, and a new SPI organization. Second, in an interview in March, 2004, the key manager at SmallSoft described and reflected on important events in the firm's quality assurance (QA) initiative from 1996 to 2004. Third, in fall 2005 we interviewed other key actors on their views of SPI technology adoption. The fourth main source was the first author's participation in an action research project with SmallSoft from fall 2003 to fall 2005; this initiative provided e-mail correspondence, documents, notes, minutes from meetings, and extensive research notes. Adding flesh to these primary data sources, we also had access to the firm's internal SPI document archive, graduate student reports from collaboration with the firm, recordings from meetings and work situations inside the firm, and interviews with research colleagues. Finally, we had extensive experiences from our personal participation in the process of adopting SPI technology at SmallSoft.

It is recognized that the direct participation in the two action research projects introduces possible bias in selecting and interpreting data. Second, the use of retrospective data from the interviews, although quite common (e.g., Denis et al. 1995; Sutton and Hargadon 1996), involves the risk that interviewees forget or rationalize what originally happened. To reduce the adverse effect of these factors, we took advantage of our access to many different complementary sources to triangulate findings (Yin 1994).

Data Analysis

Initially, we formed a historical map covering a time span of ten years (1996-2005). We viewed change processes as sequences of events, classified as either encounters or episodes (Newman and Robey 1992). Episodes are relatively stable periods of evolution that are punctuated (Peterson 1998) by compact periods of revolutionary events called encounters. Working systematically through the data sources, we identified candidate encounters by selecting events that were either mentioned in the interviews as important or that we found to have significantly impacted the adoption process. As an initial step towards a coherent case story, we mapped the identified encounters according to the chronological timeline and described them briefly together with the intermediary episodes. Subsequently, we went deeper into the data and started to analyze each encounter-episode. In doing so, we focused on SPI related activities, on their impacts on software development practices, and on activities and events in SmallSoft's environment relevant to adoption of SPI

technology. As these more detailed analyses progressed, we iterated the selection and description of encounters and episodes until a satisfactory story emerged.

Finally, to make sense of how SPI technology was adopted at SmallSoft, we identified two intrinsically related socio-technical networks. First, there was the relatively stable and powerful *production* network in which managers and software developers across SmallSoft's three departments developed new solutions in response to customer requests, primarily based on knowledge and experiences available within the firm. Second, there was the less stable and weaker *improvement* network through which a small group of different actors over time attempted to improve practices in the production-network through explication and diffusion of practices and adoption of new development technologies. These two networks thus offer complementary perspectives on SmallSoft's adoption of SPI technology. One focuses on coping with the everyday reality of software production and the other on the ongoing SPI efforts. In line with Actor Network Theory (Callon 1986; Callon and Law 1989; Latour 1987; Law 1991; Walsham 1997), each network includes actors within the firm, ways of working, current and emerging technologies, as well as relationships to forces outside the firm.

In our first analysis of the case (Tjørnehøj and Mathiassen 2008) we viewed this adoption process through the lenses of control and drift (Ciborra 2002; Ciborra et al. 2000), and—in this second analysis—through the lens of organizational improvisation (Cunha et al. 1999; Kamoche et al. 2003). Working through the encounters and data once again, we wrote up a coherent story of SPI technology adoption and of how it was shaped through a combination of management control and improvisations triggered by unpredictable events in the environment.

3.2 Process Analysis

The adoption of SPI technology in SmallSoft passed through seven encounters and subsequent episodes during the period from 1996 to 2005. This chronology is summarized in table 2.

ISO-9001-Certification

The QA effort was initiated in 1996 and successfully completed in 1998 with an ISO-9001 certificate. SmallSoft's QA system was developed by a group consisting of a department manager operating as QA coordinator and two key developers representing both management interests and engineering practices. The QA coordinator's interest was to adopt best practices firm-wide by codifying this knowledge into the QA system as mandatory procedures. Top management focused, however, on the certificate; they believed it would help promote SmallSoft as a professional software house. The QA system was implemented as an on-line library of procedures, checklists, and templates supplemented

	ISO-9001 Certification	SPI Action Learning	SPI Pilot Projects	Forming the SPI Organization	SPI Champion Exit	Learning from SPI Failure	A Grassroot Approach
	1996-1998	Spring 2000 - January 2001	Autumn 2000 - March 2001	January 2001 - June 2001	November 2002 - Spring 2004	Summer 2004	November 2005
SPI activity	Designing IS--9001 QA system	Participating in action learning program in project planning and SPI	Asessing processes, planning and designing new process. Conducting SPI pilots	Designing, negotiating and decising on new centralized SPI organization	Participating in SPI action research project. Forced to focus on sales activities which ended the SPI activities	Management realizing SPI breakdown an d starting to reflect on present and past improvement practice	Designing and implementing a SPI PIT-organization
SPI organization	An ad-hoc representative QA group (two employees and the QA coordinator)	An ad-hoc action learning group (3 key employees appointed to the education). The QA coordinator is internal sponsor	An ad-hoc action learning group (3 key employees appointed to the education). 2 production projects as pilots. Top manageement and the supervisor are members of the steering-committee	A new centralized permanent SPI group appointed. (2 from the action learning group and the QA coordinator) Ad-hoc PIT groups for each SPI initiative can be formed	Until the breakdown: The same central SPI group + the action researchers as consultants After the breakdown: No SPI organization	No SPI organization However the QA coordinator is still engaged personally	A permanenet PIT-organization is implemented. All employees participate in an improvement team. The QA coordinator is still sponsor
Immediate impact of SPI activity	Successful ISO-9000 certification & implementation of the new configuration management tool	Successful learning as new knowledge and energy was feed into the firm. No change of practices	Successful CMM assessment, knowledge of appropriate change-practice gained, new processes designed and tested	Successful design and decision on a new SPI organization Members were appointed to the SPI group and started working	The SPI champion left the firm, partly because of the breakdown of the SPI initiatives	The QA coordinator develops new understanding and views SPI practice. Renewed collaboration with researchers	Successful and enthusiastic design and implementation of the new 'grassroot' SPI organization
Impact of SPI activity in subsequent episode	Failure of implementation of the QA system	No episode (the SPI pilots were launched right after)	New knowledge of SPI and organizational change. No change of practice (see also next encounter and episode)	No change of practice. Successful evaluation of organizational change practice documented.	All SPI initiatives stop.	Elaboration on ideas and research	3 worksing PIT's. 18 employees and managers involved in the improvement network. Future success unpredictable.

Table 2: Chronology of SPI technology adoption

with an internally developed requirement, configuration, and change management tool. No permanent QA group or other quality techniques were implemented. Project managers were simply expected to comply with the QA system and all employees received introductory training in an attempt to spark the knowledge sharing and improvement of practices in SmallSoft's production-network.

After the certification, management left no doubt about the order of priority in SmallSoft as successful sales resulted in increased workloads; everyone focused entirely on producing software and serving customers. No time and energy was left to institutionalize procedures or reflect upon resulting practices; the newly formed improvement-network was largely reduced to the QA system itself that did not bring any noteworthy changes to practice. However, configuration management practices were improved leading to better customer relations, when the new tool was successfully implemented as part of the QA system; the tool was initially developed by employees in immediate response to customer dissatisfaction.

In a later analysis (spring 2001) of the impact of the QA system, the action learning group (see below) summarized these developments:

> "There is no doubt that the process leading to the ISO 9001 certificate has improved the quality in software processes. However, the focus on quality assurance and software improvement has been declining among both management and employees since then" (The action learning report).

SPI Action Learning

In spring 2000, the local university offered SmallSoft an action learning education programme (Mathiassen et al. 1999) in project management and improvement. SmallSoft*'s* management welcomed this unexpected opportunity for learning that triggered an improvised SPI initiative and thus formed a self-governing action learning group with three key employees. They hoped that their involvement in education would help grow an informal university-industry-network within the region, boost SmallSoft's image as a professional firm, and provide a useful update on software engineering knowledge. However, they kept their own focus on the production-network, responding to business and customer needs.

While attending the course at the university, the action learning group studied project management and SPI theory and engaged in discussions with teachers and fellow students from other firms. The action learning was conducted as supervised projects applying theory to support interventions in each participating firm. The action learning group identified difficulties within SmallSoft in prioritizing improvement work in relation to everyday production work as the main threat to successful SPI. As a consequence, they formally established an improvement-network as a counterweight to the production-network and as a means to more effectively involve management and colleagues in SPI activities, sharing their SPI knowledge.

SPI Pilot Projects

The action learning group agreed with management to conduct interventions according to the IDEAL-model (McFeeley 1996). In the diagnosis phase, they assessed SmallSoft*'s*

process maturity through a simplified CMM-like questionnaire focused on maturity level two: requirement management, project planning, project tracking and oversight, subcontract management, quality assurance, and configuration management.

SmallSoft failed the assessment by complying, in one extreme, with a little less that half of the norm for 'project tracking and oversight' and 'quality assurance' and, in the other extreme, with 70% for 'configuration management.' The action learning group reflected on this result:

> "It is surprising that requirement management and project planning do not score well relatively. They provide the core contracts of development projects, they have been the focus of an earlier QA initiative, the QA system contains procedures and standards for these areas and IT-based tools are available for both." (The action learning report).

The action learning group presented the result to management, who subsequently wanted to keep investments low by integrating already planned improvements on project planning. The group agreed on two interventions: (1) project planning, tracking, and oversight, and (2) quality assurance:

> "Project planning, tracking, and oversight was by management regarded as top priority, partly because the assessment and experiences from earlier, and partly because they were already preparing an experiment with a new planning and tracking procedure." (The action learning report).

The concept of 'quality meetings' was spontaneously created and agreed upon as the QA initiative at the same meeting inspired by experiences from a SPI initiative in another Danish firm:

> "Quality meetings with a project external reviewer throughout a project's lifecycle should help evaluate the state of both the process and the product." (The action learning report).

Interventions into two pilot projects were planned by the action learning group taking the management policy of no extra workload for pilots into account. The action learning group expected to gain improvement experience from the pilots among other advantages:

> "The pilots allow modification of the processes before introduction to the entire firm, if necessary. Ambassadors (the participants) are trained as a spin-off to help spread the new practices to the rest of the organization." (The action learning report).

The action learning group planned four meetings with each pilot, two of which prepared and planned the intervention while two practiced the new processes under their supervision. The introductory discussion of assessment results and participation in planning the interventions created a positive attitude. However, applying the newly designed proce-

dures and templates for project planning was a failure. The first pilot had already committed externally to a plan, so they preferred to focus on risk and stakeholder analysis instead. The other pilot planned according to the proposed procedure, but shortly thereafter, the customer unexpectedly wanted to turn the project into a flexible prototyping exercise. Both initiatives failed, simply because they lacked coordination with the stronger production-network.

The introduction of quality meetings was more successful in both pilots. A workshop introduced a coherent, flexible, and simple one-page paper based tool to support planning, conducting, and documenting the quality meetings:

> "The workshop was a success and participants from both pilots supported the quality meeting concept. The pilots especially focused on the advantages of being externally evaluated, getting external sparring and a second opinon of problems. All agreed that this would lead to higher quality of work." (The action learning report).

The first quality meeting in both pilots led to improvements in plans and products, and the participants found the practice simple, useful, and effective. The pilots were evaluated both by management and as part of the education; the results contributed to the forming of a new SPI organization for SmallSoft.

Forming the SPI Organization

While engaged in the pilots, the action learning group negotiated a plan for continued SPI with management over three meetings. As a result, management gradually adopted SPI as a way to promote SmallSoft as a professional software house, thus supporting the improvement-network. After the diagnosis in October 2000, management was introduced to SPI by the university supervisor and committed to participate in a workshop on further adoption of SPI technology in SmallSoft:

> "In an introductory presentation the supervisor listed arguments for continuous process improvements, described possible tasks, and estimated necessary resources." (The action learning report).

At the workshop a few weeks later, the action learning group provided management with a status on the pilots, an overview of all existing improvement initiatives at SmallSoft, and a tailored proposal for SPI in SmallSoft based on CMM:

> "The action learning group found 25 current and planned improvement initiatives. These were either initiatives described in strategy and plans, or so called bubblers that had been spontaneously initiated as a local initiative within a department." (The action learning report).

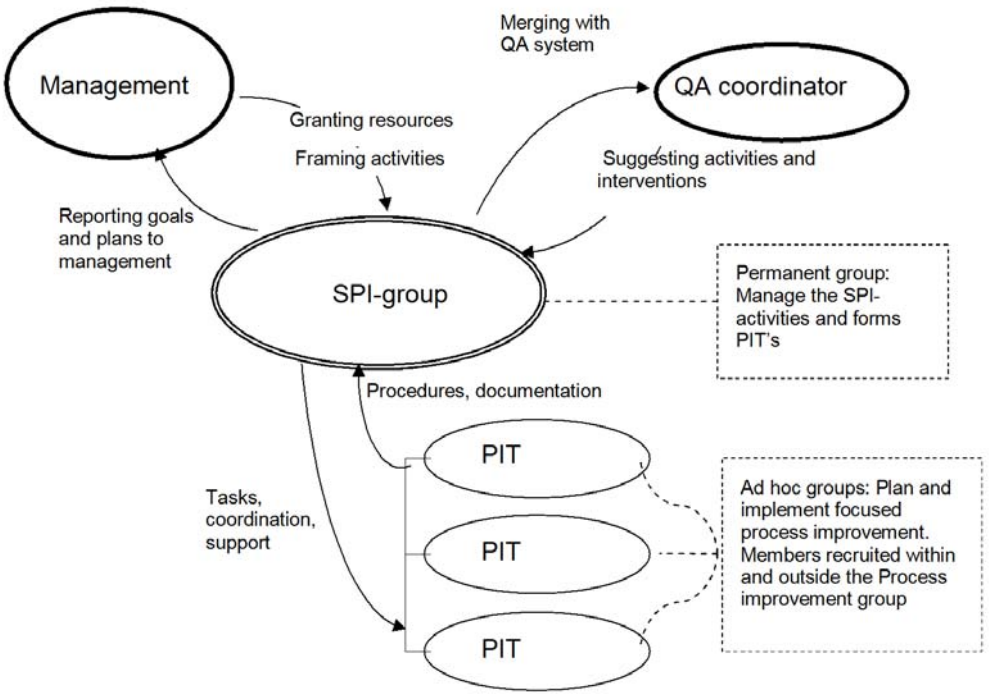

Figure 1: SPI organization based on CMM

The bubblers were a well known part of the SmallSoft culture, as employees were expected to do their best and change practices if needed. This high degree of delegation led, over time, to important improvements of software practices, improvised as local or even individual initiatives that were triggered by events or needs unplanned for.

Management approved the proposal as interesting and feasible, but asked for more detailed planning before they could commit fully. In the beginning of March 2001, when the pilots were done, the action learning group presented detailed plans for a SPI organization for SmallSoft. The group suggested limiting improvement focus to project management, but top management wanted to include software engineering and customer relations. Revising and implementing practices tested in the pilots and preparing for a new CMM assessment were the first tasks.

The SPI organization (figure 1) was centered round an SPI group coordinating and planning initiatives handled by dedicated process improvement teams (PITs) consisting of members of the SPI group and interested developers.

The decision cited in the minutes from the meeting was:

> "A SPI initiative is established in SmallSoft as described in the proposal with the noted changes. At the next staff meeting (April 2001), this initiative will be presented to all employees of the firm. Before the meeting the new SPI group will be formed and ready to work." (Minute from the meeting).

Due to the well prepared proposal, the comfortable order situation, and the positive experience with the QA system, management was very supportive. The top manager proclaimed that SmallSoft would reach CMM level three in three years even though the QA coordinator and the SPI group were skeptical.

The formation of the SPI group resulted in some experienced developers feeling undercut in their professional authority since they had not been chosen as members. The SPI group included the QA coordinator and employees with extensive SPI knowledge. All members were, however, deeply involved in the production-network and could hardly find time or energy for SPI. The group met for the first time in June 2001 and it took another meeting in August before they – six months late - presented the SPI organization at a staff meeting in November.

One member acted as SPI champion; he deeply believed in the quality meeting concept as a realistic way to improve practices and products. He pushed the concept as part of the August meeting agenda and presented a detailed and revised quality meeting process persuading the others to implement the concept in their departments straight away. He personally conducted the first quality meeting in September 2001.

In summary, while management believed SPI technology adoption was on track, this episode was characterized by low SPI activity and neglect or even hostility from developers left outside the formally established improvement-network. Even the SPI champion did not succeed in engaging the other members of the SPI group. No other developers were involved since no PITs were formed.

Priorities in the firm still favored the production-network, but the QA coordinator, having no energy left for SPI initiatives still looked for ways to empower the improvement-network. He engaged a student group from the local university in fall 2002 to analyze the ongoing SPI efforts. The students recommended decentralization of SPI based on self-improving software teams, due to the customercentered, delegated, and experimenting culture at SmallSoft. The analysis was discussed with the QA coordinator, but it had no visible influence on the improvement-network.

To sum up, the adoption of SPI technology failed and the improvement-network continued to be characterized by rather weak and heterogeneous interests.

SPI Champion Exit

Early 2003, SmallSoft entered an action research project with the local university hoping to revitalize SPI technology adoption. The researchers and the SPI group planned SPI ac-

tivities such as a best practice survey, implementation of code review procedures, and improvement of the quality meeting practice (five quality meetings were conducted during the first 6 months of 2003). The number of activities was overwhelming for the SPI group, but since the SPI champion hoped the collaboration could bring about real change, he dedicated personal energy and working hours.

During fall 2003, SmallSoft experienced a decline in the market and intensified their sales efforts. They had to downsize for the first time ever in spring 2004. The situation drained focus away from the renewed SPI effort and SmallSoft postponed or cancelled most activities. The SPI champion became increasingly frustrated and finally left the firm. The champion disagreed with management priorities and found that a strong improvement-network was crucial to the long term survival of the firm. As a result, all SPI activities stopped, but the research project leader pushed the QA coordinator to reengage or leave the project. This external pressure led to negotiations of continued collaboration.

Learning from SPI Failure

The failure of the centralized, norm-based SPI initiative forced the QA coordinator to reflect on the situation. At this point, in fall 2004, he defined the central organization as a complete failure:

> "In the beginning, we wanted a central SPI group to coordinate and manage SPI activities without doing all the work. We told people to forward their ideas for improvement to us, and then we would organize implementation in practice. It was a complete failure" (Interview with QA coordinator)

He pointed to three reasons for the failure: lack of time in the SPI group, neglect from everybody else, and the need to maintain monthly economic surplus for the firm limiting investments in innovation. The QA coordinator was, however, still committed to the improvement-network since he believed continuous improvements were necessary to sustain the firm.

He realized the dominance of the production-network over the improvement-network, and reflected on how successful improvements had been driven by production needs, thus creating overlap between the production-network and the improvement-network. Several improvements had been implemented during the relatively short lifetime of the firm, especially before the formation of the SPI group. These improvements were often initiated by developers experimenting with innovations, which subsequently spread by word of mouth (bubblers). Both the configuration management system and the QA system were initiated in this way before they were upgraded to firm level and granted resources:

> "Then I started to worry if forming a central SPI group had killed the grassroots improvements, and if nothing happened in the SPI group, what would

> then be the result? No improvements anywhere?" (Interview with QA coordinator).

The QA coordinator started to advocate for a grassroots-view of SPI as suitable for SmallSoft, in line with the earlier recommendations from the student group. Three considerations led to his conclusion: small firms cannot afford to invest in a dedicated improvement unit; the centralized SPI initiative implied increasing overhead; and a centralized SPI group wouldn't know established practices well enough to propose realistic and efficient changes.

The researchers agreed to investigate how this approach to SPI could be facilitated and implemented so the improvement-network was better in line with the production-network at SmallSoft.

A Grassroots Approach

Projects and individuals of SmallSoft had proven capable of improving practices often triggered by unforeseen changes in demands, but exploration of grassroots approaches to SPI raised the question of how local improvements spread firm wide. In December 2004, the researchers conducted a social network analysis of the improvement-network displaying very close connections within each department, but only loose connections across department borders and to top management (Nielsen and Tjørnehøj 2005), see also chapter 9. Thus, SmallSoft had a highly developed network for local improvements, but no basis for sharing knowledge and practices across departments. The researchers suggested either making SPI initiatives local to departments, thus losing the benefits from sharing knowledge, or building relations between departments to support knowledge sharing.

These insights strongly impacted a SPI strategy meeting in March 2005 by engaging two department managers (including the QA coordinator) and top management in a discussion of experiences with SPI, the QA system, and the problems and benefits of having self-governed developers and departments. In the light of a welcomed sales boom making it difficult for the new department to keep up with demands, management felt that they could learn from the two established departments to avoid well known failures. Thus, knowledge sharing between departments was given priority within SmallSoft.

One of the researchers had just learned how a well-reputed Danish firm had successfully moved to level 3 in CMM by involving all employees in PITs. This success inspired SmallSoft to form eight improvement initiatives across departments in a matrix-like PIT organization. Management decided to engage all developers in one or more PITs and, as a strong sign of management commitment, they granted man-hours to the task. Employees were assigned during summer 2005 to ensure participation from all departments and more projects in each PIT (table 3), facilitating knowledge sharing across organizational boundaries.

		PI	PI	...	PI
Name 1 Name 2 Name 3	D D D	X X	 X X	...	 X
...					
Name X Name	D D	X	X	...	 X
Name Y Name	D D	 X	 X	...	X

Table 3: The PIT matrix organization

In August 2005, the PIT organization was launched at a kickoff meeting engaging most developers in discussing the implications of the new approach. The PITs were self-organized and autonomous with respect to improvement directions, but management launched each of them according to priority.

In November 2005, three out of eight PITs were active, but working at a slower pace than expected. The PITs experienced major differences in practices between departments. Members from the established departments were expected to transfer practices to the new department, but the new department had its own traditions and preferences. Despite these challenges, the PITs made progress and arrived at acceptable solutions. Eighteen developers and parts of management were now participating in an improvement-network that seemed to grow stronger than ever during adoption of SPI technology at SmallSoft.

4 Discussion

4.1 The Role of Improvisation in SmallSoft

The aim of the follwoing analysis has been to investigate why, when, and how improvisation shaped adoption of SPI technology at SmallSoft. The analysis of improvisation in the adoption process, based on the framework in table 1 (Cunha et al. 1999; Kamoche et al. 2003), shows that improvisation played a role in most encounters except the economical breakdown. In fact, improvisations were present at SmallSoft in many different shapes with varying degrees, types, levels, and outcomes. In some encounters, actors improvised

on behalf of the organization in response to an unexpected event, but in ways that contradicted other actions in the situation.

An example of contradicting improvisation can be found in the 'ISO-9001-Certification' encounter that focuses on evolving the QA system for software development in SmallSoft. In terms of improvisation, this encounter formalized procedural memory (Moorman and Miner 1998) that in itself could hinder improvisation (Cunha et al. 1999). The reasons for introducing additional procedural memory into SmallSoft were partly trends within the software discipline but mostly management's wish to some extent to control the production-network. The culture of the production-network was dominated by autonomous domain-experts with close customer contact, mostly working in small one or two person projects. Because of the close connection to customers, these employees were constantly exposed to unexpected events and to time and performance pressures. They also generally lacked theoretical software-development knowledge and only rarely shared experiences from practice with other employees. They therefore commonly felt an immediate need to act on the events (improvise), even though they might have time to plan, search for knowledge and carry out orderly action. These micro-improvisations, as a consequence, took place on a daily basis at the individual level of the organization, as responses to emerging project challenges and were mentioned in both management and employee interviews.

The attempt to introduce a shared procedural memory of the firm was intended to eliminate or at least reduce such improvisations by implementing procedures that were more appropriate in terms of efficiency, risk, and productivity (e.g., the QA coordinators interest in the QA system). However the 'ISO-9001-Certification' episode was characterized by increasing time pressures due to a welcomed, but unexpected sales success; since the need for immediate responses seemed obvious to everybody in the production-network, the level of improvisation was increased, rather than reduced, striving to satisfy emerging customer demands. Management accepted that the new procedures were pushed into the background, thereby sanctioning contradictory actions compared to the intentions behind the ISO-9001 Certification.

The micro-improvisations described above continued to dominate the production-network culture through all ten years, ensuring flexibility, personal learning, and motivated employees, but maybe at the cost of over-improvising (not working efficient) due to addictiveness to this form of response. The benefit was that employees continued to produce what was expected, despite insufficient resources and a constantly changing environment. The micro-improvisations at SmallSoft are, on the one hand, expressions of an experimental culture with low procedural memory; on the other hand, they are also the result of management practices that provide insufficient structural support, coordination and leadership for improvisation in the production-network (Cunha et al. 1999). Leadership can be executed very differently in relation to different types of improvisation (Kamoche et al. 2003), but it plays a key role in facilitating improvisations to benefit the organization as a

whole. Coordination can be based on minimalist structures to ensure a shared feeling of urgency and an appropriate level of coherence among individual improvisational actions. Third-order controls, shared goal-setting, and deadlines are often part of such minimalist structures (Cunha et al. 1999). However, these elements were only sparsely present in relation to the micro-improvisations at SmallSoft. Deliberately choosing and utilizing these as part of management practices could have increased benefit of the improvising culture for SmallSoft.

The micro-improvisations at SmallSoft were complemented by macro-level improvisations when management or organizational actors in close collaboration with management faced unexpected events on the organizational level. These improvisations were present in the row of three encounter-episodes that started with the 'SPI Action Learning' encounter. In these cases, improvising actions were not intrinsically related to the daily activities of the production-network; instead, they unfolded at the organizational level with the economy, strategy, reputation, and political factors playing important roles in shaping them. The timeframe of macro-improvisations may be longer, but the triggering events are still characterized by being unplanned for and making actors feel a need for immediate action.

SmallSoft received an unexpected offer to join a local action learning effort. Management improvised and asked three leading employees to participate, expecting them to bring back and utilize new knowledge in the improvements of software practices within the firm. The products of this improvisation were the centralized SPI organization and learning about organizational changes and software processes; the SPI pilots were also important parts of these improvisations spreading knowledge to the organization. A minimalist structure to facilitate these improvisations was provided by the educational environment (including goal setting and exams) and by the theories taught and shared. The leadership was to some extent performed by the supervisor, but supported by and in collaboration with SmallSoft's management. These improvisations varied in degree (Weick 1998) from 'variations' over the SPI theory when shaping the new SPI organization, to 'full flesh grass roots improvisations' in the design of the new QA tool. Improvisation towards a new SPI organization as part of the SPI effort was a success in relation to SmallSoft's immediate goals; therefore it was unexpected when the following episode displayed the mismatch between organizational reality and the outcome of the improvisational process, e.g., the central SPI organization never worked. Even though this particular improvisation was guided by both appropriate minimal structural support and leadership as part of the education, the product did not fit reality (Vera and Crossan 2004). SmallSoft also improvised at this point to exploit possibilities for improvement over a longer time period, when involving the students from the local university and the researchers in the action research project.

During the encounters of the breakdown of SPI technology adoption, where the SPI champion exited and the QA coordinator reflected on the adoption of SPI technology, no

improvisations took place with regard to process improvement. All available resources were spent on sales activities in response to the problematic economic situation at SmallSoft. However, as a result of the reflections, the QA coordinator recognized that the micro-improvisations within the production-network were a main source of improvement within the firm. Moreover, he feared that the emphasis on building and enforcing a procedural memory could weaken this productive, experimental culture. Together with the researchers, he therefore improvised in the encounter 'A Grassroots Approach' to find ways to merge the grass roots approach within the production network with the need to share knowledge and procedures across the firm. This improvisation was driven by goals expressed by SmallSoft's management and based on experienced problems in the firm culture. The need for a new approach to SPI became urgent when—again—good sales in the new department happened unexpectedly, bringing the challenge related to knowledge transfer and sharing of software processes across the firm to the forefront of attention. During the management meeting, the goal was clarified and the new SPI organization was sketched, inspired by experiences from another firm.

In summary, through the ten year period SmallSoft constantly improvised to meet unexpected events at all levels of the organization. Being a small firm in a turbulent environment with a market under transformation, many events could not be foreseen and planned for. The firm culture at SmallSoft was experimental with a low level of both formal and informal procedural memory, leaving room for a high degree of improvisation (Moorman and Miner 1998). We found micro-improvisations happening at the individual level and responding to project-local events within the production-network. These were characterized by impulsiveness and hardly supported by structures and leadership. The micro improvisations were mostly uncoordinated with the overall goals of the firm and the material utilized was local, highly situated, and rarely shared across projects and departments. On the other hand, the macro level improvisations happened at the management level of the organization as a response to unexpected events in the firm's environment. At this level, we found examples of both high and low coordination and of structures and leadership that supported the improvisations.

The improvising culture and ability of SmallSoft can be seen as a great strength for a small firm in a turbulent and unpredictable world, provided that improvisations address appropriate challenges and are supported and coordinated to ensure benefits for the firm. On the other hand, improvising within SmallSoft when there is no need—because events could be planned for either through procedural memory (routine) or because there is no real time or performance pressure—can jeopardize efficient production.

4.2 Improvisational Advice on SPI

Based on our analysis of the SPI adoption process at SmallSoft and the theory on organizational improvisation, we provide advice for managers of firms that want to exploit improvisation in adoption of complex technologies.

Facilitate deliberate improvisation

The microimprovisations of SmallSoft happened rather impulsively while the macro-improvisations were more deliberate, i.e., unplanned events were evaluated to understand their potential to bring new ideas and changes to the firm and, based on this evaluation, management could deliberately choose to improvise. Facilitating such deliberate improvisations helps a firm improvise when appropriate, i.e., when the situation calls for it. We thus advise managers to identify and analyze unexpected events to determine whether they are suitable triggers for improvisation in the technology adoption process (Cunha et al. 1999).

Provide explicit support structures

The improvisation in the 'Action learning' and 'SPI pilots' encounters was guided by explicit coordination mechanisms supported by the (minimal) structures of the formal education. The improvisation thus became manageable and coherent, aiming at a well known and shared goal that included both learning and change. Even though the product, i.e., the formalized SPI organization, was rather unsuccessful, the outcome was mostly positive, that is, in learning, flexibility (change indeed), and motivation. Coordination can be provided formally, but often explicitly shared goal setting (sometimes involving management) is sufficient. Common minimal structures are deadlines (Cunha et al. 1999), tools, and theories that decrease the risks of improvisation. In adoption processes, technology often plays an important role as a minimal structure and an implicit coordination mechanism. Management is advised to plan such coordination mechanisms and minimal structures and to provide them during improvisations. Management should also ensure that structures stay minimal to leave as much room for improvising as possible (Cunha et al. 1999).

Exercise leadership

Explicit leadership of improvisations was sparse at SmallSoft, except at micro-improvisation level, since improvising individually on unexpected events takes leadership and responsible acting. Explicit leadership of the improvisations at macro level could have sparked innovation in SmallSoft. The leadership style recommended when improvising is one of supportive and collective turn taking because this invites and encourages each

member to give contributions to the ongoing improvisations, thus ensuring that all knowledge, skills and resources add to the solutions (Cunha et al. 1999). This points to a need for collective improvisations at both micro- and macro-level to increase learning, innovation and knowledge-sharing. Management is often the formal leader of the improvising group, and should, on the one hand, perform servant leadership to create room for the leadership within the group thus reducing pressure on the members; on the other hand, it should serve the organizational objectives through clear goals and other minimal structures.

Cultivate improvisations

The experimental culture at SmallSoft facilitated improvisation that sustained the innovative capability of the firm, but in some situations they improvised even when they did not have to. It is important to cultivate knowledge, values, and practices to develop and sustain the improvisational capability of the firm to stay innovative, at the same time keeping production efficient; it is important to set standards to avoid improvising in routine situations or other situations that can actually be planned for and carried out according to procedural memory. Such behaviors can lead to inefficient work practices and low quality. Management should hence strengthen and secure sharing of procedural memory throughout the firm as SmallSoft did, especially in the last grassroots approach encounter.

5 Conclusion

As indicated in the literature review; organizational improvisation is a rather unexplored field, specifically within IS and technology adoption. We have investigated the role of improvisation in a longitudinal study reporting from a ten year period adopting SPI technology in SmallSoft, thus adding significantly to the empirical evidence in the field. The strength of the study, that is, its being longitudinal, is also the source of its main limitations. Much of the data is retrospective either through the interviews or through our own involvement in the case. The amount and complexity of the data also calls for both analysis and interpretations. We have triangulated between many data sources and research to reduce the effect of this, achieving a positive result.

We found two significant different levels and types of improvisations that persistently interacted, often uncoordinated, throughout the whole period, shaping the innovation of the firm. Micro-improvisations mostly individually performed and characterized by being variations over personal practices and skills when facing new and unexpected project challenges. Macro-improvisations on the organizational level, where planning and action still converge, but hastiness of the action is less obvious. Instead strategy, the economy, reputation and political factors play an important role in shaping the action. As the formal leaders of macro-improvisations management can serve both the improvising group and

the organizational objectives through goal setting, while it is harder to secure that the micro-improvisations are in line. We find that further research on improvisation has to take into account the possibility of different and opposing improvisations in one organization.

Based on this study, we advise managers how to enhance an appropriate improvisational culture of their organization through deliberate leadership, thus balancing learning, knowledge sharing, innovation and efficient production when adopting SPI technology.

References

Balla, K., T. Bemelmans, R. Kusters and J. Trienekens (2001). Quality Through Managed Improvement and Measurement (QMIM): Towards a phased development and implementation of a quality management system for a software company. *Software Quality Journal*, 9(3): 177-193.

Barrett, F. J. (1998). Creativity and Improvisation in Jazz and Organizations: Implications for organizational learning. *Organization Science*, 9(5): 605.

Batista, J. and A. D. D. Figueiredo (2000). SPI in a Very Small Team: A case with CMM. *Software Process: Improvement and Practice*, 5(4): 243-250.

Brodman, J. G. and D. L. Johnson (1994). What Small Businesses and Small Organizations say about the CMM. In: *16th International Conference on Software Engineering*. Sorrento, Italy.

Brouse, P. S. and R. T. Buys (1999). Affordable ways to improve application development. *IT Professional*, 1(4): 47-52.

Callon, M. (1986). Some Elements of a Sociology of Translation: Domestication of Scallops and the Fishermen of St Brieuc Bay. In: *Power, Action and Belief: A new sociology of knowledge?* J. Law, ed. London, Routledge & Kegan Paul: 196-223.

Callon, M. and J. Law (1989). On the Construction of Socio Technical Networks: Content and context revisited. *Knowledge and Society* (8): 57-83.

Carnegie Mellon, S. E. I. (2005) "Process Maturity Profile (http://www.sei.cmu.edu/appraisal-program/profile/pdf/SW-CMM/2006marSwCMM.pdf).

Casey, V. and I. Richardson (2004). A Practical Application of the IDEAL Model. *Software Process: Improvement and Practice*, 9(3): 123-132.

Chelariu, C., W. J. Johnston and L. Young (2002). Learning to Improvise, Improvising to Learn: A process of responding to complex environments. *Journal of Business Research*, 55(2): 141.

Cho, S., L. Mathiassen and A. Nilsson (2006). *Event-Based Actor Network Analysis of IT-Based Change.* Georgia State University.

Ciborra, C. (1999). Notes on Improvisations and Time in Organizations. *Accounting, Management and Information Technologies*, 9: 77-94.

Ciborra, C. (2002). *The Labyrinths of Information: Challenging the wisdom of systems*. New York, Oxford University Press Inc.

Ciborra, C. U., K. Braa, A. Cordella, B. Dahlbom, A. Failla and O. Hanseth (2000). *From Control to Drift: The Dynamics of Corporate Information Infrastructures*, Oxford University Press.

CMMI Production Team (2000). SCAMPISM, V1.0 Standard CMMISM Assessment Method for Process Improvement: Method Description, Version 1.0. Pittsburgh, Carnegie Mellon University.

CMMI Production Team (2002). CMMISM for Systems Engineering/Software Engineering/ Integrated Product and Process Development/Supplier Sourcing, Version 1.1, Staged Representation (CMMI-SE/SW/IPPD/SS, V1.1, Staged), Technical Report SEI.

Coleman, G. and R. Verbruggen (1998). A Quality Software Process for Rapid Application Development. *Software Quality Journal*, 7(2): 107-122.

Crossan, M. (1997). Improvise to Innovate. *Ivey Business Journal*: 36.

Crossan, M. and M. Sorrenti (1997). Making Sense of Improvisation. *Advances in Strategic Management*, 14: 155-180.

Crossan, M. M. (1998). Improvisation in Action. *Organization Science*, 9(5): 593-599.

Crossan, M. M., H. W. Lane, R. E. White and L. Klus (1996). The improvising Organization: Where planning meets opportunity. *Organizational Dynamics*, 24(4): 20.

Cunha, M. P. e., J. V. d. Cunha and K. Kamoche (1999). Organizational Improvisation: what, when, how and why. *International Journal of Management Reviews*, 1(3): 299.

Dion, R. (1992). Elements of a Process-Improvement Program. *IEEE Software*, 9(4): 83-85.

Hansen, B., J. Rose and G. Tjørnehøj (2004). Prescription, Description, Reflection: The shape of the software process improvement field. In: *UK Association of Information Systems Conference*. P. Powell and K. Grant, eds. Glasgow, Scotland, Glasgow Caledonian University.

Horvat, R. V., I. Rozman and J. Györkös (2000). Managing the Complexity of SPI in Small Companies. *Software Process: Improvement and practice*, 5(1): 45-54.

Humphrey, W. S., T. R. Snyder and R. R. Willis (1991). Software Process Improvement at Hughes Aircraft. *IEEE Software*, 8(4): 11-23.

Kamoche, K., M. P. e. Cunha and J. V. d. Cunha (2003). Towards a Theory of Organizational Improvisation: Looking beyond the jazz metaphor. *Journal of Management Studies*, 40(8): 2023-2051.

Kautz, K. H. (1999). Making Sense of Measurement for Small Organizations. *IEEE Software*, 16(2).

Kautz, K. H., H. W. Hansen and K. Thaysen (2000). Applying and Adjusting a Software Process Improvement Model in Practice: The use of the IDEAL model in a small software enterprise. In: *International Conference on Software Engineering*. Limerick, Ireland.

Kelly, D. P. and B. Culleton (1999). Process Improvement for Small Organizations. *Computer*, 32(10): 41-+.

Kilpi, T. (1998). Product Management Challenge to Software Change Process: Preliminary results from three SMEs experiment. *Software Process Improvement and Practice*, 3(3): 165-175.

Kuvaja, P., J. Palo and A. Bicego (1999). TAPISTRY - A software process improvement approach tailored for small enterprises. *Software Quality Journal*, 8(2): 149-156.

Latour, B. (1987). *Science in Action: How to follow scientists and engineers through society*. Cambridge, Massachusetts, Harvard University Press.

Law, J. (1991). Introduction: Monsters, Machines, and Socio Technical Relations. In: *A Sociology of Monsters: Essays of power, technology and domination*. J. Law, ed. London, Routledge.

Leung, H. K. N. and T. C. F. Yuen (2001). A Process Framework for Small Projects. *Software Process: Improvement and Practice*, 6(2): 67-83.

Mathiassen, L. (2002). Collaborative Practice Research. *Information, Technology & People*, 15(4).

Mathiassen, L., F. Borum and J. S. Pedersen (1999). Developing Managerial Skills in IT Organizations-A case study based on action learning. *Journal of Strategic Information Systems*(8).

Mathiassen, L., J. Pries-Heje and O. Ngwenyama, eds. (2002). *Improving Software Organizations: From principles to practice*. Reading, MA, Addison-Wesley.

McFeeley, B. (1996). IDEAL: A User's Guide for Software Process Improvement. Pittsburgh, USA, Software Engineering Institute.

McKay, J. and P. Marshall (2001). The Dual Imperatives of Action Research *Information Technology & People*, 14(1): 46 - 59

Miner, A. S., C. Moorman and P. Bassoff (1997). *Organizational improvisation in new product development*, Marketing Science Institute, MSI.

Mirvis, P. H. (1998). Practice Improvisation. *Organization Science*, 9(5): 586-592.

Moorman, C. and A. S. Miner (1995). *Walking the tightrope: improvisation and information use in new product development*, Marketing Science Institute, MSI.

Moorman, C. and A. S. Miner (1998). Organizational improvisation and organizational memory *The Academy of Management Review*, 23(4): 698-.

Newman, M. and D. Robey (1992). A Social Process Model of User-Analyst Relationship. *MIS Quarterly*, 16(2): 249-266.

Ngwenyama, O. and P. A. Nielsen (2003). Competing Values in Software Process Improvement: An Assumption Analysis of CMM from an Organizational Culture Perspective. *IEEE Transactions on Engineering Management*, 50(1): 100-112.

Nielsen, P. A. and G. Tjørnehøj (2005). Mapping Social Networks in SPI. In: *IFIP 8.6 Conference: Business Agility and IT Diffusion*. Atlanta, GA, USA,, Springer Verlag, 2005.

Orlikowski, W. J. (1996). Improvising Organizational Transformation over Time: A Situated Change Perspective. *Information Systems Research* 7(1): 63-92.

Orlikowski, W. J. and J. D. Hofman (1997). An Improvisational Model for Change Management: The Case of Groupware Technologies. *Sloan Management Review*, 38(2): 11.

Paulk, M. C. (1998). Using the Software CMM in small organizations. In: *Sixteenth Annual Pacific Northwest Software Quality Conference Joint ASQ Software Division's Eighth International Conference on Software Quality* PNSQC.

Peplowski, K. (1998). The Process of Improvisation. . *Organization Science*, 9(5): 560-561.

Peterson, M. F. (1998). Embedded organizational events: Units of process in organization science. *Organization Science,* 9: 16-33.

Pettigrew, A. M. (1990). Longitudinal Field Research on Change: Theory and practice. *Organizational Science: Longitudinal Field Research Methods for Studying Processes of Organizational Change* 1(3): 267-292.

Richardson, I. (2001). Software Process Matrix: a Small Company SPI Model. *Software Process Improvement and Practice*, 6(3): 157-165.

Rodenbach, E., F. van Latum and R. van Solingen (2000). SPI - A Guarantee for Success? A Reality Story from Industry. In: *Product Focused Software Process Improvement*. Berlin, Springer-Verlag Berlin. 1840: 216-231.

Steel, A. P. C. (2004). Low-rigour, Rapid Software Process Assessments for Small Software Development Firms In: *2004 Australian Software Engineering Conference*.

Susman, G. I. and R. D. Evered (1978). An Assessment of the Scientific Merits of Action Research. *Administrative Science Quarterly*, 23(4): 582.

Saastamoinen, I. and M. Tukiainen (2004). Software Process Improvement in Small and Medium Sized Software Enterprises in Eastern Finland: A State-of-the-Practice Study. In: *Software Process Improvement: 11th European Conference, EuroSPI 2004*. Trondheim, Norway, Springer Verlag.

Tjørnehøj, G. and L. Mathiassen (2008). Between Control and Drift: Negotiating Improvement in a Small Software Firm. *Information Technology & People,* 21(1).

Truffley, A., B. Grove and G. McNair (2004). SPICE for Small Organizations. *Software Process Improvement and Practice*, 9(1): 23-31.

Varkoi, T., T. Makinen and H. Jaakkola (1999). Process Improvement Priorities in Small Software Companies. In: *Technology and Innovation Management. PICMET '99. Portland International Conference on Management of Engineering and Technology.*

Vera, D. and M. Crossan (2004). Theatrical Improvisation: Lessons for Organizations. *Organization Studies*, 25(5): 727-749.

Villalon, J., G. C. Agustin, T. S. F. Gilabert, A. D. Seco, L. G. Sanchez and M. P. Cota (2002). Experiences in the application of software process improvement in SMES. *Software Quality Journal*, 10(3): 261-273.

Walsham, G. (1993). *Interpreting Information Systems in Organizations*. Chichester, Wiley.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, 4(2): 74-81.

Walsham, G. (1997). Actor-network theory and IS research: current status and future prospects. In: *Information Systems and Qualitative Research*. A. S. Lee, Liebenau, J. and DeGross, J.I., ed., London, Chapman and Hall,: 466-480.

Ward, R. P., M. E. Fayad and M. Laitinen (2001). Software Process Improvement in the Small - A Small Software Development Company's Most Difficult Challenge: Changing Processes to Match Changing Circumstances. *Communications of the ACM*, 44(4): 105-107.

Weick, K. E. (1993). Organizational redesign as improvisation. In: *Organizational Change and Redesign*. G. P. Huber and W. H. Glick, eds., New York, Oxford University Press 346-379.

Weick, K. E. (1998). Improvisation as a Mindset for Organizational Analysis. *Organization Science*, 9(5): 543.

Wilkie, F. G., D. McFall and F. McCaffery (2005). An Evaluation of CMMI Process Areas for Small- to Medium-Sized Software Development Organizations. *Software Process Improvement and Practice*, 10(2): 189-201.

Wohlwend, H. and S. Rosenbaum (1994). Schlumberger's Software Improvement Program. *IEEE Transactions on Software Engineering*, 20(11): 833-839.

Aaen, I., J. Arendt, L. Mathiassen and O. Ngwenyama (2001). A Conceptual MAP of Software Process Improvement. *Scandinavian Journal of Information Systems*, 13.

10

The Road To High Maturity

How the first Danish company reached CMMI level 5 in 100 months

Jan Pries-Heje , Jacob Nørbjerg Ivan Aaen and Thomas Elisberg

1 Introduction

Since the early 1990s many organizations have embraced the idea of Software Process Capability and Software Process Improvement (SPI). A documented maturity level is becoming increasingly important for organizations opting for contracts in various areas; e.g. the US Department of Defense contractor for the Joint Strike Fighter expects at least a CMMI (Capability Maturity Model Integrated; cf. Chrissis et al. 2003) maturity level of 3 or 4, depending on the type of project; further, in Denmark the Ministry of Science, Technology and Innovation has developed a model to measure maturity that they suggest has become an integrated part of a standard contract between supplier and public customers.

SPI is, however, a very complex, resource demanding, and long-term process. On average, it takes 18 months to move up one maturity level , and many have reported a very high failure rate. For example, in a study of a large number of organizations, Goldenson and Herbsleb (1995) reported that in SPI 26% agreed that "nothing much has changed" and 49% declared themselves to be disillusioned.

This chapter, however, describes and analyses how a company did succeed in moving from level 1 to level 5. The process was very successful in the sense that the company managed to reach the goal it set out to reach. Section 2 views improving maturity in essence as an organizational change process and ends with choosing a simple organizational change

model as our theoretical lens for categorizing findings. Section 3 summarizes what maturity and CMMI is about. Section 4 presents our research method, including data collection and data analysis. Section 5 describes the case study of how the Danish software house Systematic progressed from level 1 to level 5, measured using CMMI. Section 6 sums up our findings before the conclusion in section 7.

2 Organisationel Change

Our starting point is that SPI is in essence an organizational change process, that is, the processes in an organization, and the behavior and interaction of people, groups, projects and, in fact, the whole organization. Many authors have written about organizational change from different perspectives including psychology, sociology and business. Contributions to organizational change have been built on empirical work in a wide variety of organizations. There have been both descriptive accounts of change and normative models that aim to guide the change process.

In terms of the historical accounts of change, three different schools of organizational thinking have provided metaphors to describe the nature of the organization and organizational change.

The first school—and the oldest approach to organizational design and change - dates back to the end of the nineteenth century where Frederick Taylor, Henri Fayol and Max Weber pioneered organizational theory as we know it today. In this perspective, an organization is seen as a production system where it is possible to optimize the system's efficiency and effectiveness. Thus, organizational change is about optimizing planning through observation, experimentation, calculation and analysis.

In the 1930s and 1940s the second school challenged the classical view of organizations to provide a new perspective. In relation to change, this perspective is characterized by the belief that organizations are co-operative, social systems rather than mechanical ones, where people seek to meet their emotional needs (Burnes 1996; Borum 1995). So the metaphor for an organization is a (large) group of people with an organizational culture and visible communication and interaction processes between them.

The third school of thought has been called the political-emergent perspective (Burnes 1996; Borum 1995), characterized by the belief that organizational change is shaped by the values, interests, commitments and power-struggles of individuals and groups. They compete amongst themselves for power and resources; there are differences of opinion and of values, conflicts of priorities and of goals (Handy 2005).

If we now turn from describing to prescribing, the most prominent example of a prescriptive model for organizational change is Kurt Lewin's (1951) three-step model from 1951: Unfreeze - Move - Freeze.

To *unfreeze*, the change agent—that is the person or group responsible for making change happen—has to make the organization receptive to change. The individual must

realize that there is a need for change: typically, by identifying a problem of relevance to the individual. This first step is called unfreezing because if the change agent does not take the time to create organizational receptivity, the organization will behave like a block of ice; it will naturally resist change.

To *move*, the change agent typically proposes a solution to the relevant problem that was identified during the unfreezing process. In this stage there will be factors that promote the change and others that work as barriers to change. A very simple change tactic is then to support the promoting factors and suppress the barriers.

The third step is to *freeze,* that is, to make sure that the change becomes a permanent part of how the organization works. You freeze water thereby ensuring a more permanent shape. Likewise, one 'freezes' the organization to make change permanent.

The strength of the unfreeze-move-freeze model is its simplicity. It gives clear and simple prescriptive guidelines for implementing change. One needs to go through all three steps and in the prescribed order. Weick and Quinn (1999) ascertain that the Lewin model has been "surprisingly durable over the years" and that the model "continues to be a generic recipe for organizational development."

Another widespread normative model was developed by John P. Kotter (1996). He recommended eight steps in an organizational change process: (1) establish a sense of urgency, (2) build support, (3) develop a change vision, (4) communicate the change vision, (5) empower and enable action, (6) generate short-term wins, (7) consolidate and re-vitalize change, and finally (8) anchor a new approach in culture. Steps 1 to 4 in this model are more or less equal to Lewin's 'unfreeze,' steps 5 to 7 relate to moving the organization, and step 8 is the same as what Lewin calls 'freeze.' This confirms the rather bold statement by Hendry (1996, p. 624) where he purports that one can "scratch any account of creating and managing change and the idea that change is a three-stage process which necessarily begins with a process of unfreezing will not be far below the surface. Indeed it has been said that the whole theory of change is reducible to this one idea of Kurt Lewin's."

If we look for guidance on organizational change in the SPI literature, Humphrey (1989)—one of the founding fathers of CMM and CMMI which is the most widespread maturity model used for SPI—also reuses the Lewin model.

Thus, there appears to be a considerable number of arguments in favor of choosing the Lewin change model as our theoretical lens when analyzing the change process involving maturity improvement.

3 Maturity and CMMI

The Capability Maturity Model (CMM) is a framework characterizing a 5-step path for software process improvement (Paulk et al. 1995). The path describes key processes at each of five levels. The description includes a number of goals at each level. An organization has to meet the goals at one level to reach the next. For example, to go from the basic

level 1 where behavior is characterized by being ad-hoc and intuitive to level 2, you need to achieve the goals incorporated in six key process areas: requirements management, subcontractor management, project planning, project tracking, quality assurance, and configuration management.

The CMM became so popular that a large number of other models using the same 5-step path were invented. For example, People-CMM, Integrated Product Development CMM, Systems Acquisition CMM (Cooper and Fisher 2002), Testing Maturity Model (cf. Burnstein 2002), and several others. Finally, a large number of the CMM-models were summoned in CMM-integrated—or just CMMI. The major difference between CMM and CMMI is that Systems Engineering is included. That means that there are more key processes, making it more useful in companies that are not only developing software but also combining software with, for example, mechanics, electronics or other kinds of hardware (cf. Ahern et al. 2001, Chrissis et al. 2003, CMMI Product Team 2002).

If we use the terminology of Lewin (1951) as argued above, we can characterize the maturity improvement process using CMMI as a series of unfreeze-move-freeze cycles.

Each cycle begins with an assessment according to a maturity model such as the CMMI. The assessment identifies the goal of the change, as well as the changes needed to reach that level; i.e., reaching the next higher maturity level and the changes in the organization's software processes required to do so. The purpose of the moving phase, then, is to define and introduce new or improved software processes. The change process ends when the new processes have been disseminated in the organization and a new assessment confirms that the organization has reached the next maturity level. This also marks the beginning of the next change cycle - unless you have reached the highest level, at which time you just maintain the change you have achieved.

4 Research Method

The case study approach was adopted as the main strategy of inquiry. Thus, this study can best be categorized as an intrinsic case study with the purpose of telling the story and coming to an understanding of the challenges in normative software process improvement (Stake 2000).

The long term collaboration between the case company and the research group has made it possible to study the company from the beginning of the software process improvement effort to the culmination with the CMMI level 5 certificate eight years later.

Three of the authors of this chapter first came into the Danish software house Systematic in 1997 as part of a large research undertaking (cf. Mathiassen et al. 2001). Over the years we gathered comprehensive documentation, conducted numerous interviews, reviewed material, participated in meetings, etc. The fourth author was introduced to Systematic in 2003 and worked as an intern in the company as part of an action research undertaking in 2004-05.

The data was collected applying a multi-method design using historical data as well as new data. After Systematic reached CMMI level 5 in November 2005, we gathered all the material we had from eight years of research and structured it according to the levels in CMMI: level 1 to level 2, 2-to-3, 3-to-4 and 4-to-5. The collected material was used as basis for a search conference in 2006. Based on action research and similar to a focus group, search conference participants "bring the whole system in the room" to exchange views and learn from one another. Participants share observations, engage in collaborative analysis, and logically test discoveries in an interactive debate. Search conferences produce data and findings coincidently (Levine et al. 2002).

Like other forms of action research, search conferences are expedient, providing rapid and useful outcomes, while to some extent sacrificing the scientific goals of replicability and proven validity. We organized the search conference with a common start where 4 researchers and 5 SPI practitioners were in the room. After an hour we broke into four subgroups, each with 1 researcher and 1 practitioner focusing on obtaining each of the levels from 2 to 5. Then after 2 hours we met in common again and summarized all the findings. We recorded the whole search conference and transcribed key parts. The transcripts were used for further analysis and coding of the data. In the case study below all citations stem from the search conference transcripts.

After the first round of analysis additional interviews were conducted in November 2007. This extra round of data gathering and analysis added a management perspective in that the Systematic CEO participated in a 5-hour interview that emphasized the role and responsibility of top management. Furthermore the interview with three people in November 2007 focused on the effect and outcome of having obtained the level 5 certification.

For coding we used the theoretical lens that the Lewin's (1951) change model provided. That gave us four main categories of observations for each transition to a higher maturity level: the unfreeze, the process, promoters and barriers, and the effect and benefits. Finally, a comparative analysis of the four categories across the four transition phases was carried out to identify patterns across maturity levels and to highlight the different challenges that the company faced over time as their maturity increased.

5 Case Study

Systematic was established in 1985 in Aarhus, Denmark. SSE is the largest privately-owned software company in Denmark with top management holding the controlling interest. Systematic is an International Systems Company with headquarters in Denmark and legal companies in Great Britain, the United States and Finland. The core business areas of Systematic are the Defense sector, the Healthcare IT sector and the Government sector.

Systematic has expertise in large complex and mission critical IT systems, and are supplier for customers in more than 30 countries worldwide of software solutions, integrated systems, own products, training and consultancy. Specifically, Systematic develops a product suite named Iris that offers state-of-the-art services within military messaging and interoperability. Systematic are specialists in application integration, data security and project management. They do not have very many different customers, but the relatively few that they have are typically large and have long-standing relations with Systematic. Over many years Systematic has measured customer satisfaction; the most recent measure was 4.1 on a scale from 1 to 5 (5 being very satisfied).

From spring 2005 - 2007 Systematic has grown from 350 to 440 employees, of which 50 are in the UK and the USA. 3 out of 4 systems engineers hold an MSc or PhD degree; Systematic also has high employee satisfaction and low employee turnover. The yearly investment in competence development is equivalent to 9% of the payroll.

Below is given an overall timeline for the improvement process that has taken place at Systematic. After the timeline, a detailed case description for each of the four transitions from maturity level 1- 5 is presented.

- 1992 Systematic receives ISO 9000 certification
- 1997 Systematic decides to use the CMM to improve
- 2000 A Bootstrap assessment (Kuvaja et al. 1994) shows that Systematic is at level 2
- 2001 Systematic uses a Balanced Score card first time - Use continues and becomes an integrated part of reaching level 4
- 2002 Systematic passes a formal CMM level 3 certification
- 2002 Systematic decides to change from CMM to CMMI (see section 3 for a short explanation of the difference)
- 2003 A second Bootstrap assessment measures Systematic to be at level 3,25
- 2004 Systematic passes a formal CMMI level 4 certification
- 2005 Systematic passes a formal CMMI level 5 certification

5.1 At The End of The Road Looking Back

The overall outcome of going down the road of maturity and reaching level 5 after eight years are better processes, improved quality and less rework. In other words Systematic sees the outcome of the improvement process as a clear success. “We are much more at-

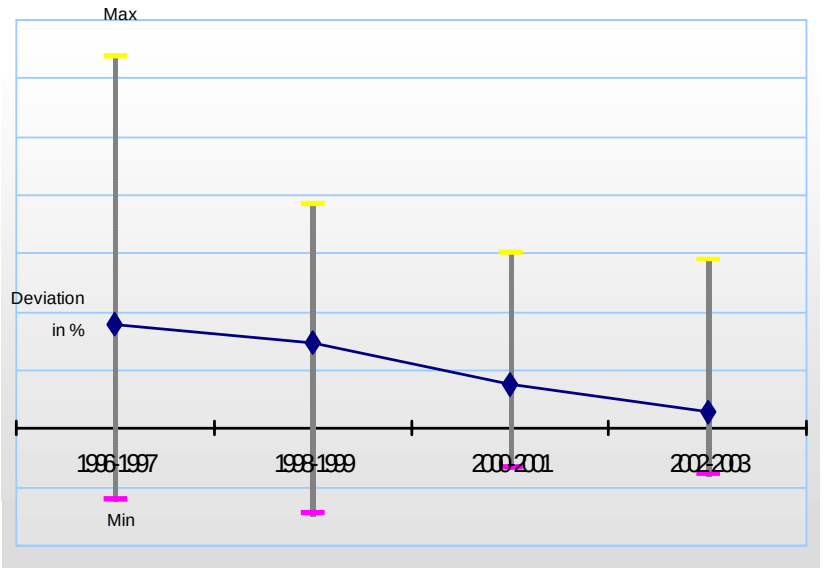

Figure 1: Deviation between actual and estimated effort [% in hours]

tractive to customers both nationally and internationally as well as for employees and potential due to our CMMI activities and results" says the CEO.

If we look at better processes we can take project management as an example. An important part of project management is the ability to estimate. In figure X1 we have shown how estimation precision developed from 1996-97 where Systematic decided for the first time to use CMM for improvement and until 2002-2003 where level 3 was reached (source: Systematic Videnregnskab 2004).

Thus we see that the estimation ability have improved dramatically since 1996. In year 2002/2003 the deviation between estimated and actual effort has reduced to less than 10%; that is less than one fifth of what the deviation was originally. Furthermore the variation in the ability to estimate was cut in half.

Furthermore—and related to that—the ability to deliver on time improved as well. Here are the percentages of deliveries on time for five years:

- 82% in 2002
- 90% in 2003
- 89% in 2004
- 88% in 2005
- 90% in 2006

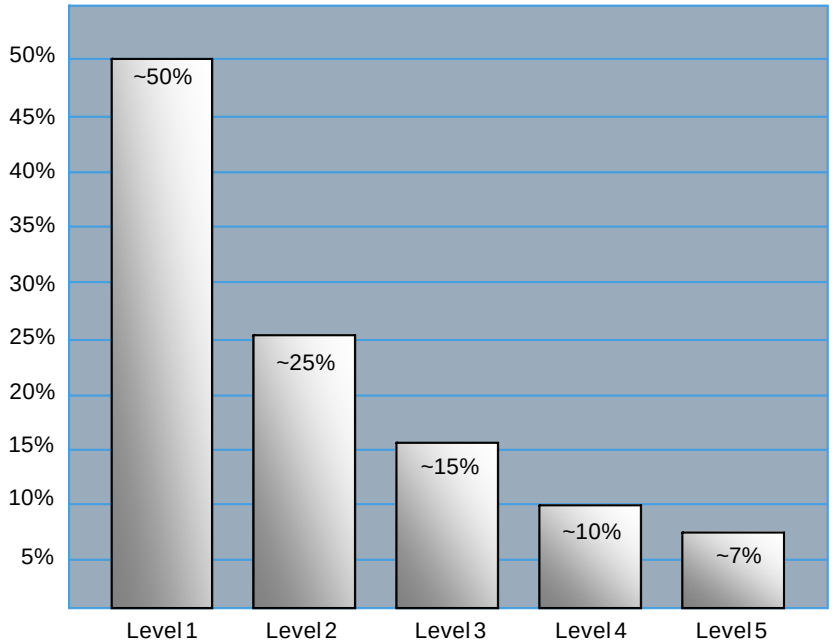

Figure 2: Published levels of rework at different maturity levels

The reason for delays was either estimation defects or customers defining requirements too close to deadlines. Both things have now improved considerably.

Improved quality and less rework is illustrated by figure 2 and figure 3.

Figure 2 shows published levels of rework at maturity level 1 to 5 (numbers from Krasner and Houston (1998) and Krasner (2001)). The Systematic CEO—when interviewed in November 2007—recalls an earlier gathering (at a low maturity level) of all developers within Systematic where he was thinking: "30% to 40% of these people are not developing software to be used by our customers they are just reworking what others have done!".

From figure 3 we can see that Systematic has been under the 7% rework level that could have been expected at level 5 (cf. figure 2) ever since level 5 was reached in November 2005. Furthermore we can see that the level of rework has continued on a downward trend in the year and a half following.

5.2 From Level 1 to Level 2

In 1992, Systematic's quality assurance system was certified in accordance with ISO 9001 and the military standards to become AQAP 110 and MIL 150, respectively. In the long term, however, these standards would be insufficient to meet both customers' increasing

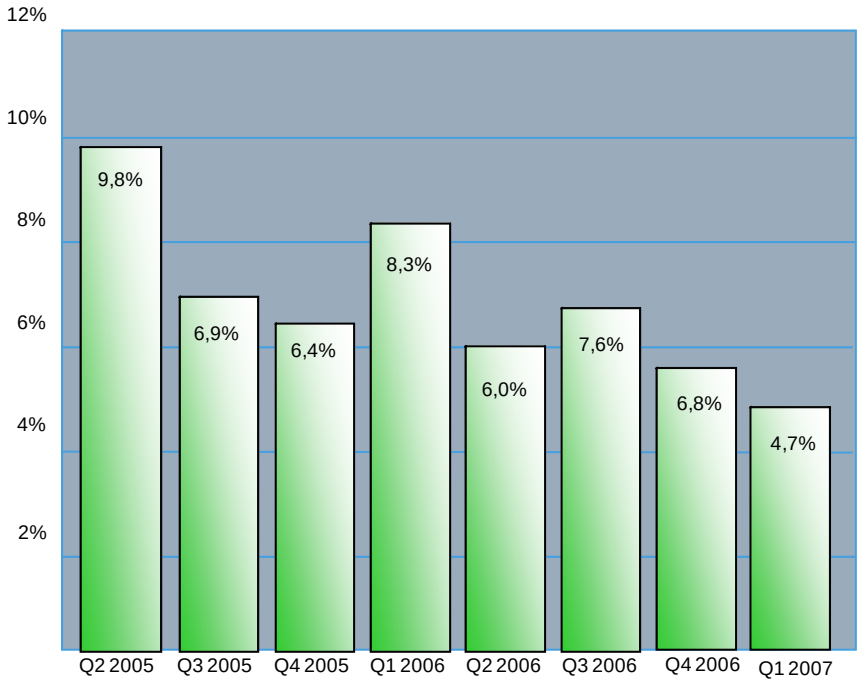

Figure 3: Measured levels of rework in Systematic 2005-2007

demands for quality assurance and internal requirements on the efficiency and predictability of the development process. Thus, in 1997, Systematic decided to use the CMM as the overall process framework for the company and to embark on an SPI effort aiming for CMM Level 3 within 3 years. For market reasons, the objective and major challenge of the SPI effort was to pursue CMM improvements while at the same time maintaining the quality assurance system as compliant with ISO 9001 and AQAP 110/150 and still run an effective and competitive business.

The "Frozen" Setting When It All Started

The management group's goals for the SPI project were twofold: (1) to establish sound software processes that would increase project productivity, quality, and scheduling predictability, and (2) to move to CMM level 3 directly. "It was not interesting in itself to have a level 2 certificate on paper," said the CEO.

The second goal was motivated by the expectation that U.S. defense industry customers would in time require all suppliers to be at CMM level 3 or higher. In short, the major challenge that Systematic faced was that a part of their market was requesting a

CMM certification, and an expectation in Systematic that this request would spread to Denmark as well.

When asked about other reasons for starting out on the road to higher maturity the CEO said:

> "First of all we wanted to distinguish ourselves from the crowd. It has always frustrated me that anyone can start out in a garage and immediately become a software house competing in the market place. We wanted to be and be seen as professional; thereby standing out from the crowd. We wanted—and still want—to make a difference."

From unfreeze to move

Initially, the person who had led the earlier ISO certification was asked to head the SPI effort. This person was very well respected in the company for his competence and ability to accomplish changes. As he was unable to allocate the time required for the SPI effort, the then quality manager soon replaced him. The quality manager was not able to adapt the maturity model to the organization and vice versa, and was therefore replaced by a newly employed software engineer. Many practitioners saw the software engineer as an outsider offering academic solutions to practical problems and did not welcome his suggestions.

A number of working groups were chartered to propose solutions for each of the CMM Level 2 key process areas. However, senior management deemed many of their suggestions insufficient, excessive, or impractical and thus decided not to release them for implementation. "The QA manager at that time lacked respect in the organization and did not have the ability to communicate" said the CEO. It became clear that Quality Assurance work (including the ISO certification) was very different from what was needed to mature.

After rework on the solutions, the next challenge was to use them in practical projects. Due to major variations in project management styles across the company, which is how things are expected to be at level 1 and 2, it proved difficult to implement at the project level. To overcome the difficulties, a major training program for project managers was initiated. Experienced outside consultants—from the Danish consultancy company Delta—were brought in to conduct the training. Included in the training were new ways of working so that the training became instrumental for the rollout. The training program also led to improved networking among project managers in the company.

The training program enabled a shift in responsibility where project managers became responsible for developing solutions, and the SPI group became responsible for the rollout. After establishing this division of responsibility, the new way of working became much more effective. Process action teams were formed to produce final process descriptions for a so-called Business Manual consisting of all the processes to be performed by

projects; it was, in fact, a company standard. Rollout of the Business Manual was done at one of the monthly 'all-hands meetings' for all employees.

After publication of the Business Manual, project managers nevertheless still complained about the processes. People from the SPI group assisted in using the processes, but for a while it was acceptable for a project manager to criticize and choose not to use a process. To push process utilization, the SPI group adopted self-assessments and established a wall of fame for all projects. This wall of fame demonstrated the openness that Systematic wanted to pursue; "Customers came around and saw it and they were quite interested" we were told. Further the wall of fame promoted a competitive but friendly spirit among projects.

A major challenge for the SPI effort was to find a balance between large and small software projects. Configuration management and requirements management were two areas where processes differed greatly, yet developing two sets of processes was not feasible. The dilemma therefore was to strike a balance with one common process for all projects. "Looking back I can see that we could have used more time on communication," the CEO reflects.

Barriers and promoters in the move stage

In retrospect, the Systematic search conference participants concluded that in the first half of the three-year period (from level 1 to level 2) the SPI group suffered from severe and interlinked problems:

- Too few people with too little time in the SPI effort
- High SPI staff turnover
- Weak SPI experience and reputation in the company

Establishing a set of metrics for planning and oversight was a challenge in a company where only basic measurements like time used and delivery time was part of daily routines. This barrier affected every key process area addressed.

Inhomogeneous project management competencies formed an important impediment for process rollout. After publishing the Business Manual, the projects experienced a severe shortage in the time assigned for process implementation. Thus, process implementation vied for priority with customer-related project assignments. On top of that, the company did not use fixed working hours. More work—i.e., for process rollout—then simply meant working longer hours. Projects often depended highly on individuals and delegation of responsibilities; consequently, making more time for rollout was rarely an option. "We learned the hard way that we had to add hours in the projects for improvement - but it was not until level 3 that we really became skilful in it" told the CEO.

The organization was split between two goals: Customer-orientation and process improvement. An effective communication of process improvement goals was lacking during that period.

In 1998 TeraQuest—an international consultancy firm specializing in process improvement—was hired in. TeraQuest organized a seminar in May 1998 for the 10-12 top and middle managers which resulted in a common vision for improvement, more clarity and agreement on SPI goals, as well as a number of concrete future scenarios for Systematic. This seminar was crucial in overcoming the above-mentioned barriers for the SPI effort. This seminar developed commitment to SPI and initiated agreement on goals in the management group; from there, this commitment and agreement spread to the other levels in the organization: "There was this familiar consultancy-effect... One crucial thing is WHO says something. [This consultant] was very good at communicating and was recognized as competent, and that was something that mattered" (Interview with SPI Manager).

With the renewed commitment from the seminar came adequate resources and status for SPI as well as higher priority for improvement efforts in an organization where customer demands and requirements used to be 'king'—that is, had highest priority. It also changed so many employee attitudes to SPI that the CEO asked one of the most esteemed project managers in the organization to head the SPI effort as his main and only responsibility.

A condition for the SPI effort was that existing certificates (ISO, AQAP) must not be jeopardized when changing processes. This concern meant that process descriptions and the Business Manual were to be compliant with preexisting descriptions.

In the early spring of 2000 the consultancy company Delta performed a Bootstrap assessment at Systematic which showed that maturity level 2 had been reached.

The benefits and effects of achieving level 2

Early in the SPI effort a number of expected benefits for achieving maturity level 2 were identified, but during the effort focus became more long-term towards a desirable future scenario. Immediately following the successful level 2 assessment the company aimed for level 3.

The three main effects of the move to CMM Level 2 were: (1) that project management was formalized within Systematic and all project managers came to have a common vocabulary, (2) that a common understanding of and vision for SPI was developed across managers, the SPI group, and projects, and (3) that the SPI group was stabilized, managed, and staffed.

"On the other hand, the main group of users of the processes—the developers in Systematic—was hardly affected. 'When will something happen for us?' was a question often heard."

As early as late 1998 the SPI group developed a return-of-investment (ROI) projection based on figures from the SPI literature on costs and benefits related to achieving CMM Levels. These somewhat theoretical calculations were part of what convinced the management group about the value of doing SPI and helped ensure resources for the effort.

A more indirect effect of the SPI effort was that Systematic started reporting on employee satisfaction, intellectual capital, and customer satisfaction. These publicized reports came out in 1998 but it was not until 2002 that they had any major influence on or information from the SPI effort.

Reflection on the experience of moving from level 1 to 2

When we asked at the search conference and in November 2007 about things that Systematic or the SPI group should have done differently, we had three answers. The first was that if they were to start over today from maturity level 1, they would emphasize vision much more: describe and communicate clearer what the company wants, and why. Building a shared and strong vision—including mission, strategy, and values—earlier would have made the SPI effort markedly easier.

The second answer was that the company would be fully aware that level 2 are for the few (project managers mainly). Thus it should be communicated to the developers that it was okay that they did not see or experience any changes.

The third answer was that the company would organize more pilot projects where highly motivated and inspired people produced results that could inspire the remainder of the organization.

5.3 From Level 2 to Level 3

No formal CMM level 2 assessment was carried out in Systematic. Instead, a European Bootstrap assessment (Kuvaja et al. 1994) was used to show that Systematic had achieved level 2 early in the year 2000. The assessment result was strongly influenced by the completion of the first version of a Business Manual. However, it can be questioned whether level 2 in Bootstrap are transmittable to level 2 in CMM. One of the consultants from TeraQuest actually phrased this question when she conducted an informal evaluation of what was needed to reach level 3.

The frozen setting when initiating the level 2-to-3 process

After level 2 was formally reached and results from the Bootstrap assessment published Systematic experienced a period that could be called 'post-performance exhaustion.' This is a well-known phenomenon in the world of sports that you after a high performance will

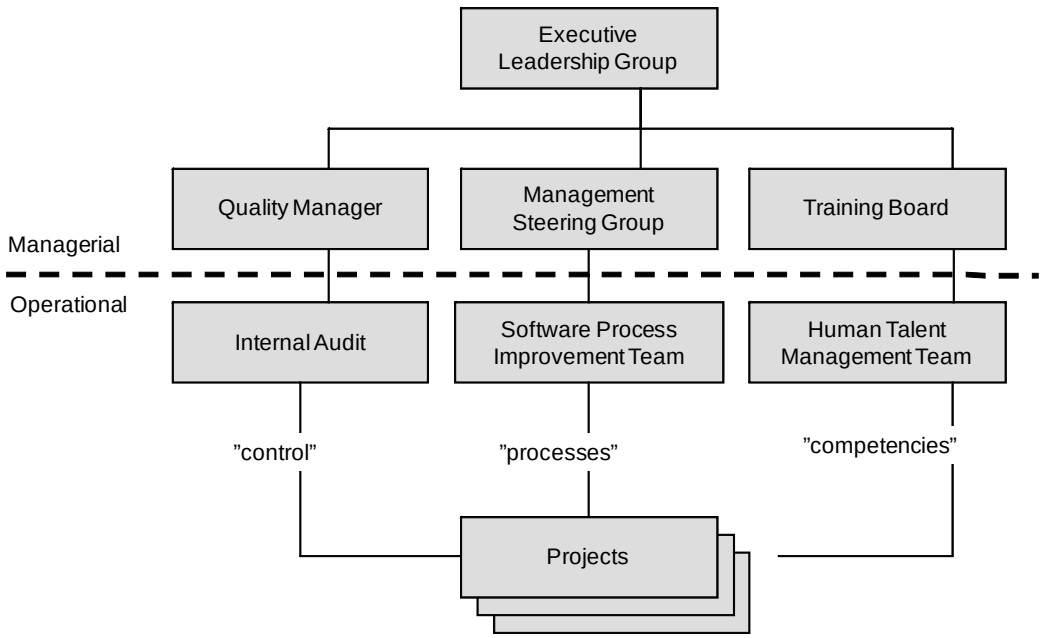

Figure 4: The new improvement organization at Systematic

experience a period of extraordinary fatigue (cf. Schillings et al. 2003). This post-performance exhaustion has happened every time a new maturity level has been reached told the Systematic CEO, and it has lasted - typically - between three and six months.

Some months after the level 2 assessment in the early spring of 2000, the new SPI manager concluded "Either we drive into a dead end, or we staff the SPI project with people that have the competence to lift this," he said. As a result a third project manager was assigned to the SPI project. Furthermore, funding for continued use of external consultants was found. And with these two things in place—manager and money—Systematic was back on the road to level 3.

The move from level 2-to-3

The major change that happened here was a total reorganization of the improvement effort. As told above a new manager for SPI was in place. More importantly a new organization was put in place. One of experiences from reaching level 2 was that training was extremely important. Therefore a 'Training Board' was established. Furthermore a 'Management Steering Committee' was set up with top management at then end of the table. The third part was the old QA organization still in place and working. In figure X4 the new organization for SPI is shown.

During the summer of 2000 the teams released a Business Manual that described level 2 and most level 3 processes. After release, the teams were disbanded. Project managers were expected to adjust their practices to the new manual by, among other things, reworking project plans on on-going projects to comply with the new Business Manual.

The new full-time SPI project manager was recruited internally in late 2000, and during the first half of 2001 *fourteen* additional full-time persons were allocated to the SPI group. By the end of 2000 the Business Manual included descriptions of all CMM level 3 processes and in addition to this processes for training and measurement. There was no significant changes to the manual before the level 3 assessment.

Barriers and promoters in the move

Moving to level 3 was, however, a major organizational change process that required significant efforts, particularly in three areas:

- To understand the nature and potential benefits of measurements
- To develop appropriate support tools
- To change the process mind-set of project managers and developers

First, regarding measurements and metrics, the general understanding was that these were primarily related to levels 4 and 5. However, Systematic decided to build a measurement system. In relation to that the SPI project group, therefore, spent considerable time internally and with software developers discussing what to measure, how to measure, and the potential value of measurements.

> "There was, at the time, no measurement culture and no understanding of their potential value ... What to measure? If you measure, how do you ensure that you have measured correctly? And if you measure correctly, how do you interpret the results; what are the benefits? So it was a giant task, first to make [the SPI group] understand [this] and then for [the SPI group] to get the rest of the house to understand it." (SPI project manager)

When looking back in November 2007 the three participants interpreted what happened here in the following way: "The SPI group had to learn how to communicate and sell things internally; that was not easy, and it took time to learn it."

Second, the SPI group developed and acquired support tools for the processes as described in the Business Manual. One important part of this was the so-called 'PDP Common' (PDP = Projects Defined Process). There was one PDP Common and it worked well for large projects. Small projects (1-5 people) found it often to cumbersome.

Existing tools that supported individual project processes were, at the same time, gradually phased out. The new tools were seen as a way both to implement common processes, a common work culture and as a support for reliable measurements and analysis:

> "We came from this environment, where the projects largely decided [how to work]. Changing to the new situation, with a shared database where we would collect information across the house ... then we would all have to use the same tools.... It was crucial to us ... to become able to aggregate measures at the company level. And that requires similar process to generate the measurements." (SPI overall manager)

Most of the support tools were developed in-house in Systematic. The SPI project manager estimated that defining the measurement program, developing support tools for measurement as well as other processes, and convincing developers and project managers about their usefulness took about eight months.

Third, the SPI project group had to build support and motivate many of the project managers. Some project managers were enthusiastic towards the SPI idea and would work independently with improvements, while some others were indifferent, but required much help and support; however, very few was against it. A major task was, therefore, to convince the last two groups of the value and benefits of common processes, measurements, etc.

Tailoring support tools to Systematic's processes was one means employed in this regard, but the SPI project group would also use the SPI enthusiasts and their experiences to create a more positive attitude towards SPI: "I consciously used them against each other - I simply placed them together. We spent a long time identifying areas, where [the enthusiasts] had actually created value. Where the projects had achieved something that they could not have achieved otherwise. And then tell the good stories about it" (SPI project manager).

Towards the end of 2000 there was considerable resistance towards SPI among project managers. This resistance was partly a reaction towards the pressure put on them by the SPI program: "The projects ... must meet their dead-lines .... But when you order them to spend 100-200 hours to revise the plans for a project that meets current company standards—then you will not find much understanding" (SPI project manager). The project managers were also reluctant to accept a change in focus from the quality of the product they delivered, to the processes they used to create the product. Finally, understanding the need for measurements, and establishing a reliable and trusted measurement program—along with support tools - were major challenges

The initial pressure to adapt existing projects to the Business Manual contributed, interestingly enough, to some project managers' positive attitude towards SPI: "[Here] we saw the first project managers that realized that they were after all not as different from each other as they believed. ... They were under hard pressure to revise their project plans

[according to the Business Manual], and they could only complete the revision if they were compliant," said the SPI overall manager at the search conference.

Finally, the use of outside consultants helped the SPI group to overcome resistance, and to identify and address key issues in the change process: Systematic became assessed at CMM level 3 on 27 March 2002. The assessment was carried out by the international consultancy company TeraQuest.

The benefit and effects of reaching level 3

Reaching the goal—level 3—was in itself a major benefit that created a feeling of success throughout the company. The SPI project manager, when interviewed specifically about his experiences with moving from level 2-to-3, identified further benefits in the areas of *customers, processes,* and *staff*.

From a customer point of view, Systematic became a more professional organization with higher estimation accuracy for both time, quality and money. Internally, the streamlined and standardized processes increased staff mobility between projects, as well as the developers' ability to work independently without guidance and immediate supervision. The requirements for staff also increased, as new employees had to be familiar with Systematic's terms and practices, and just as importantly, accept the discipline of a level 3 organization.

From a purely economic point of view, the move to level 3 represented a huge investment, however, with 14 full-time employees in the SPI group in addition to the time and effort of the rest of the development staff.

There are some Systematic measurements of the effect of moving from level 2 to 3 as shown earlier in figure 1 and 3.

Last but not least the CEO pointed out that it was the work towards level 3 that build the basis for the continuously improving organization that Systematic is today.

Experiences and reflections on going from level 2-to-3

The SPI project had unconditional support from Systematic's top management and a clearly defined goal. Both contributed significantly to the perceived success. It took a while, however, for the organization to realize the magnitude and importance of the organizational changes involved in the move from level 2 to 3. This change process should have been addressed more explicitly and at an earlier stage.

After analysis at the search conference (academics and practitioners together), we concluded that the main lessons to be learned from this process were understanding the importance of total top management commitment.

5.4 From Level 3 to Level 4

After passing level 3 in the early spring of 2002 Systematic experience post-performance exhaustion once again. In fact, nothing really happened in relation to CMM and CMMI from Easter 2002 until 6 months later.

The setting when starting towards level 4

In parallel with reaching level 3 Balanced ScoreCard (BSC) came in. BSC was originated by Kaplan and Norton (1996) mainly as a reaction to the weaknesses of traditional financial measures. BSC is not just a measurement model but is meant as a model that enables an organization to clarify its vision and strategy and to translate them into action, both being iterative and rational. Process improvement is related because it is framed and fostered by BSC.

The first version of a BSC in Systematic was developed in 2001 where a professor from Aarhus Business School was invited into the organization. He convinced Systematic that they could benefit from developing their own BSC. Thus, in the intermediate period between passing level 3 and actively going for level 4 the first Systematic Balanced Scorecard was finalized and it was decided by Systematic top management to spread the BSC out to the projects.

The first pilot project to bring BSC down to the projects happened in 2002 (a year after the first BSC for 'the company' Systematic was made). In retrospect, the SPI Project Manager realized that these first efforts were wrongly designed: projects were handed the company BSC with company goals. Projects were then supposed to formulate their own goals based on company goals: "This is what we want to do in the house—how will you support that?" This approach did not work. It required a lot of effort from the projects, and the result coming out of it was really poor in that the formulated goals could not be used at all.

Before this effort was improved, it was decided by top management that Systematic should go for level 5 (with level 4 being an intermediate step on the way to 5). And BSC was seen as having perfect synergy with the aim for high maturity.

At this point in time it was also decided to go for a CMMI certification causing Systematic to translate their existing CMM-based Business Manual and PDP common to CMMI terms. The main reason for the decision was that the Software Engineering Institute in Pittsburgh at this point in time—2003—very clearly stipulated that they planned to sunset CMM shortly after (source: www.sei.cmu.edu in 2003).

After the decision was made in Systematic to go for a CMMI level 5 certification, some time was needed to familiarize and understand what constituted level 4 (and later level 5). In parallel with this, the Balanced Scorecard received a lot of attention by management, the SPI group, and the projects. Since BSC was about defining measures and CM-

MI level 4 about measurement, it seemed obvious to combine the improvement efforts. Systematic decided to integrate BSC and CMMI in their effort to reach CMMI level 4.

In the spring of 2003—April - May—a draft plan for reaching level 4 was written by the SPI group where the decision of combining BSC and CMMI was reflected upon in detail (source: The document itself). The SPI representatives at Systematic then started to integrate it and subsequently implemented it in the organization. This combined concept was built during the late spring of 2003.

After the summer of 2003 the SPI unit was ready to train people in quantitative project management. January, February and March 2004 were dedicated to the actual use of the new processes, especially how to define and measure it. April 4th 2004, Systematic was certified CMMI level 4.

The move from level 3-to-4

As for level 3, the Business Manual and PDP common were main vehicles for describing how the projects were to behave and perform when working at level 4. The implementation process was centrally managed through the new organizational structure. Central management was needed to deal with complexity inherent in CMMI level 4. Top management in Systematic gave the SPI group a mandate to work from. Many things were tried in pilot projects. Overall the SPI group together with the Management Steering Group decided how things should work, and the Training Board ensured that developers and project managers had the training needed.

We asked at the search conference whether this centrally managed approach gave birth to any resistance. The answer was that very few people objected. In fact many projects were at this point mature enough to give good response to SPI function.

As mentioned earlier, the very first implementation of BSC was not very successful. Approximately 500 hours were used in all the Systematic projects together, but in reality the result was an inadequate reflection of the organizational BSC goals.

> "Today I don't think it was surprising that this first BSC effort in the projects failed," said the SPI project manager. "Because we didn't formulate requirements to the projects. We just gave them the company goals and asked how they supported that? So all in all this effort required hundreds of hours in each project but the investment did not pay off."

Then in the annual management strategy seminar in 2003 in which all top level and a number of middle level managers in Systematic participated, the concept of a Project BSC was further developed. The idea was that the SPI project had to read the organizational balance scorecard and transform it to the project level. So the SPI function took on the responsibility of facilitating the translation process. "We formulate what the organizational

goals means for a typical project in the house; formulated as requirements," as the SPI project manager articulated it.

Promoters and barriers

The Systematic people in the search conference were convinced that the primary motivation in going from level 3-to-4 was the inclusion of BSC. It turned out that the synergy expected between CMMI level 4 and BSC actually materialized. Thus BSC was a major promoter here.

The first barrier that Systematic encountered was that some of the elements contained in reaching level 4 were so difficult to understand that even after trying for several months, it did not result in enough of an overview to write a plan. The SPI manager said: "Some of these level 4 things are so difficult to understand that you cannot sit down and write a complete plan."

However, it became even more difficult to diffuse the knowledge of level 4 actions and succeed in having Systematic developers and projects adopt it. "The difficult thing in this is not to understand the model. The difficult thing is to make it work in practice; to make it operational. And there is only one thing to do: Keep on trying when you do not succeed the first time," as the SPI Project manager told us. "On top of all this it also needs to give business value. All-in-all that was not an easy task," added the CEO.

The second barrier that Systematic ran into concerned the measurement system. At an early point in time stakeholders—including the SPI manager—formulated a vision of building a kind of measurement data warehouse. Thereby allowing many measurements to be carried out automatically in a discrete fashion. For example, this vision included all documents and programs to be measured automatically every night. The idea was to have as little human involvement in the measurement program as possible.

The measurement data warehouse, however, required quite some time and effort - up to 5 man years in total—to make it work. "The big challenge was clean data. You never get the truth. There is always something that can be better" the SPI project manager said.

The benefit and effects

The main benefit of reaching level 4 was a change of culture to a belief in measurements. Instead of using the data warehouse as originally envisioned, the approach that was used was first to decide what key performance indicators Systematic should have, and then ask the projects for the data and let them be responsible for the quality of the data.

The second benefit mentioned in our interviews at Systematic was that BSC actually ended up working as an effective mechanism to align business goals at the organizational levels with concrete goals in the projects.

When asked about the effects of reaching level 4, the interviewees grasped the intellectual capital report from 2004, and pointed to figure 2 where it can be seen that the deviation in percent between actual and estimated hours had gone dramatically down; and that Systematic in the period reported (February 2003 - January 2004) delivered 89% of all milestones on time. Both are measures that are extremely important to Systematic customers.

Experiences to remember from the level 3-to-4 move

The interviewees did not mention many things that they would have done differently. In retrospect, the SPI project manager called the vision of automatic measurement and data warehouse the biggest mistake in the period where Systematic went from level 3 to level 4. He said: "I think we emphasized a technical solution way too much, and did not put sufficient weight on data quality and what it actually was that we collected. Of course you need some degree of automated measuring but I believe that you need to look very seriously at data you collect and at what is the real NEED for measurement data."

On the other hand the CEO and middle-manager in November 2007 claimed that automatic measurement probably was "what was needed at the time to change the culture." So in light of that it was neither a failure nor too expensive, but an important lesson learned.

In fact the CEO pointed out two very positive effects of automatic measurement and the change of culture: Transparency into and among projects as well as increased predictability. For example the automatic measurement system generated reports with traffic lights (red - yellow - green) for all the BSC measures in each and every project. This made it very easy for management to grasp problems at an early point and take preventive measures. "We know what is happening in the projects," said the CEO.

However, it is also a fact that the advanced measurement data warehouse that was built in 2002-04 never realized the benefits originally expected (i.e., totally automated data gathering and as little human involvement as possible).

Finally, when directly asked, the SPI project manager mentioned that the whole organizational change process probably became too much what the SPI-organization wanted, without enough room and time for feedback: "Systematic is always looking forward towards the vision and the next level. We could have spend more time to consolidate," he said.

5.5 From Level 4 to Level 5

The setting for the final move - From level 4-to-5

Level 5 of the CMMI model includes two additional processes: Causal Analysis and Resolution (CAR) and Organizational Innovation and Deployment (OID). A substantial part of the organizational structures that was meant to sustain the level 5 processes had been established at level 4.

In 2004, that is, 18 months prior to the commencement of the level 5 focus, the SPI group at Systematic outlined the organizational implementation of these two processes to ensure a reliable and consistent transition to level 5. Apart from the initial design of the organizational tailoring of these processes, the work on CAR and OID had been suspended until level 4 was fully implemented and externally appraised. However, the organizational mechanisms for sustaining the two processes had been prepared. One of the central mechanisms for implementing the CAR and OID processes was Knowledge Networks.

Knowledge Networks is an organizational structure that acts as a meeting place for different knowledgeable professionals in the systems development process. The SPI group at Systematic had used inspiration from the team model in Microsoft Solution Framework to identify six areas around which to organize the core of the Knowledge Networks: Program Management, Development, Test, Release Management, User Experience and Product Management. A seventh knowledge network, Architecture, was added. As part of the CAR and OID processes, the networks had the responsibility for furthering professional knowledge and ensuring process optimization.

The initial schedule for obtaining a level 5 organization was rather aggressive. The work on level 5 was commenced immediately after the level 4 certification on April 4, 2004. In November 2004 Systematic decided to postpone the date to ensure that the knowledge networks mechanism was anchored properly in the organization. Further reasons for postponing the date were to gain sufficient time to train employees to the new work processes, to generate enough evidence for the organizational implementation, and to demonstrate model compliance. Hence, after careful consideration of pros and cons, the appraisal date was deliberately postponed from 5 May 2005 to 11 November 2005.

The move from level 4-to-5

The transition from level 4-to-5 was characterized by the SPI group as being considerably simpler than the previous maturity levels. The strategic decision to aim for a level 5 was taken after the level 3 certification was acquired. The organization was therefore determined to continue to level 5 immediately after the level 4 certification was acquired.

As the overall SPI Manager reported, "We commenced the work on acquiring the level 5 certification immediately after the level 4 certification was attained. There was no

weakening in terms of effort in the SPI unit. However, in general there was a fall off in the organization," meaning that support for maturity and CMMI for a period waned. This is one more example of post-performance exhaustion.

To overcome the certification fatigue in the organization, the SPI group maintained a low profile for a few months during which time they planned the organizational implementation of the two level five processes. Based on the experiences gained from the level 3-to-4 move, a detailed communication strategy was prepared to communicate the general plan, including objectives and the outlined implementation process. The following months the SPI group spent time on preparing a brochure especially targeting employees, describing the vision as well as the upcoming organizational changes: "We spent a lot of time outlining and designing a brochure communicating what we were aiming at. This vision brochure was a huge success. First of all we [the SPI unit] had time to identify the actual objectives and communicate our vision to the organization. We had become more mature in this sense."

At the search conference the entire transition process was characterized as smooth. The overall SPI group manager gave two explanations for this: first the SPI group had become better at managing changes because they were familiar with this by that time. Second, the organization had experienced a shift in culture towards an acceptance of continuous change. Due to a constant focus on improving existing processes employees and managers in Systematic, all expected that processes or practices would not be constant. This made it much easier to manage change.

"We [the SPI unit] focus on the objective of the change process, not who is involved in the process ...We do SPI the good way; the way that works," as the overall SPI group manager expressed it.

Promoters and barriers

The smooth transition process from level 4-to-5 may give the impression that no barriers existed in this phase. However, some were identified at the search conference although they had shifted in nature. While the barriers at the earlier maturity levels dealt with organizational issues and later in the process centered on the model itself, the identified barriers at level 5 were few and of a different nature:

> "Certain traditional barriers emerge when implementing the high maturity levels: Our consultants are not able to help us in these matters. You have to come up with all the ideas yourself. Only sparse accessible empirical work exists targeting the higher maturity levels." (Overall SPI group manager)

The barriers now dealt with external issues and the lack of experience in other companies that Systematic needed to cooperate with—customers, suppliers, partners, etc. This proved to be a challenge for the SPI group, mainly because they had to come up with the

entire concept for the organizational tailoring of the concept by themselves with no inspiration from other high level companies.

Although the move from level 4-to-5 was perceived to be considerably less challenging than any of the other transitions in the CMMI model, it took time to design mechanisms tailored to the organization that honored the process requirements regarding continuous improvement and innovation. Acknowledging that the initial plan was very optimistic, the SPI group realized that the speed of change was very high if not too high: "The only trouble I can come to think about is that we acknowledge that our change tempo was very high," said one of the SPI project managers.

If the knowledge networks were to prove to be an asset and sustain the organizational implementation of the final level of the CMMI model, time was needed to attune the mechanism to the organization. This hurdle was self inflicted and was dealt with accordingly by postponing the final appraisal date, as sketched in the timeline.

Benefit and effects

In terms of benefits, Systematic capitalized significantly from the huge investment in software process improvement. The immediate benefits were of managerial character. The overall SPI group manager said—when reflecting over what should happen after reaching CMMI level 5—"Now it is time to for us to focus on how to capitalize on this investment. In practice this implies an optimization and increased efficiency of the existing processes."

In the organization the SPI group experienced general agreement among people that the process quality had increased significantly over the past decade. Some citations from Systematic employees shortly after the appraisal in 2005 illustrate this:

- "The amount of overtime has been reduced and people are more satisfied"
- "Compared to 3-5 years ago, projects are better at estimating and planning"
- "We either follow the documented process or we improve it"
- "Process descriptions are not static rules set in stone they must support our business agility and speak to users in their language"
- "People can work by themselves without consulting senior colleagues and it is easy to move between projects"
- "Formal definition of roles makes it easier for Project Managers to delegate"

These statements are supported by the metrics that was established at level 3 - 5 as illustrated by figure 1, 2 and 3. The two level 5 processes also proved to be effective in detecting defects in the process and understanding the root-cause of these defects.

However, the greatest benefit was linked to the integration of knowledge networks as a core mechanism for sustaining the level 5 processes. Over the last year (2004-05) knowledge networks had evolved to be a central organizational structure with many responsibilities. And in November 2007 knowledge networks had now taken ownership of all Systematic processes. "It works," told the CEO. Thus the role of the SPI function today is mainly to act as a secretariat for the knowledge networks. So what was very centrally driven for example from level 3 to level 4 is now more or less decentralized, in November 2007.

> "There is no doubt that we now got a mechanism to implement improvements across the organization that is very successful. This will always be present in the organization."

The general opinion is that in terms of business the level 5 certification has had a major impact on customers, with employees, and in the society in general. As an example the CEO showed us a piece from the editor-in-chief of Computerworld (Steinmark 2004) saying: "Only a few companies in Europe are at the same level as Systematic. And we find no other Danish company at the same level ... Apparently Systematic have the caliber to be an IT guiding star..."

Thus one clear effect is the increased attention from the professional community. The level 5 certification has made Systematic the center of attention for both media and professional networks. Systematic has gained a leading position within the process quality community and has a strong brand. This has, for example, made Systematic a popular keynote at conferences.

6 Summarizing Findings

We have described and analyzed how the Danish company Systematic succeeded in moving from level 1 to level 5. The process was successful in the sense that the company managed to reach the goal it set out to reach.

We have aimed to identify the nature of the unfreeze - move - freeze process as it unfolded, using the Lewin (1951) as our theoretical lens for our coding and analysis.

In the table below we provide an overview of our findings.

The following things are worth noting on the differences between the four transitions—moves—from one level to another.

At first, the development projects did not feel any need for process changes. With Systematic being the frontrunner company in Denmark in SPI from level 2 onwards, it was difficult to seek inspiration elsewhere on how to balance gradual and major process changes in a Danish business environment.

Initially, the organization was not focused specifically on process benefits. Becoming more professional, and recognized as an international player as well as obtaining the

L1-L2	*L1-L2*	*L2-L3*	*L3-L4*	*L4-L5*
The setting at un-freeze (the problems they were facing)	Major customer expected CMM certification Need to distinguish Systematic from the crowd - be seen as professionals	Post-performance exhaustion	Discussion in Systematic whether to continue the march towards level 4 and 5 CMM was being sun-set. Led to CMMI decision	Level 4 just a step on the way to level 5 Need for realizing benefits in organization
People responsible for the actual "move" process	Initially hero-based. Later SQA-based. Finally based on dedicated SPI-staff and working groups	New organization with Mgmt. Steering Committee and Training Board + SPI group and, outside consultants	Same as L2-L3, plus university professor (for BSC)	Same as L2-L3, plus to some extent recently established knowledge networks
Promoters in the 'move' process	Training programs for project managers	Training and competence development for all developers. Project managers as champions	Combining CMMI with Balanced Scorecard (BSC)	Establishing knowledge networks Communicating vision for level 5 through brochure
Barriers to the "move" process	Lack of SPI time and competence High SPI staff turnover Weak SPI reputation	Resistance to measurements Appropriate support tools	Understanding CMMI - both SPI and employees	No external sources for inspiration (due to few level 5 experiences worldwide)
Benefit and effects after re-freeze	CMM understood and broadly accepted	Process discipline Estimation precision	Aligning organizational and project goals using BSC as the key mechanism - translated by SPI More customers	Optimization and increased efficiency of the existing processes Considerable less rework Successful mechanism for implementing improvements across the organization

Table 1: Overview of analysis findings using the Lewin (1951 unfreeze - move - freeze model as our theoretical lens

CMM certificate was the main goals for a long time. From level 2 and onwards more focus was on improved quality, less rework, and satisfied customers.

ISO and AQAP initially were constraints on the emerging CMM process. It was realized that the "control and policing" (as the CEO expressed it) perspective needed for QA didn't work for SPI. A continuous improvement culture was needed instead.

The SPI organization underwent major changes on the way from level 1 to level 2. After that, the SPI organization was structurally stable, consolidated, and growing.

On the way to level 3, the sun-setting of CMM and introduction of CMMI required substantial training and communication. In fact Systematic developed a whole new organizational structure to fulfill this requirement.

A primarily centrally driven approach to process improvement was used throughout the SPI effort. However, after reaching level 5 responsibilities for processes and improvement was decentralized to knowledge networks.

At first, measurements and other kinds of feedback did not play any major role in managing the SPI effort. Self-assessments were introduced underway to level 2. From level 2 onwards measurement played an ever-increasing role. Feedback mechanisms proved hard to implement due to the lack of a measurement tradition. Eventually, consistent and trusted measurements came in place and supported the SPI effort from level 3 onwards.

Initially, the SPI effort process rollout tended to be large scale via publication of the first version of the Business Manual. Later process changes were implemented more gradually.

Commitment among developers was lacking at first. This related to the fact that moving from level 1 to 2 didn't provide anything specifically to the developers; but the project managers were in focus here. When moving to level 2 and later to level 3: a wall-of-fame, persuasion and a combination of project managers as champions were useful in building commitment.

After reaching every maturity level—2, 3, 4 and 5—a period of post-performance exhaustion was experienced.

As the process initially was developed separate from the development projects, process rollout proved difficult. From level 3 onwards process development and use became more integrated and process implementation more gradual. Late in the SPI effort, the SPI group perceived the high speed of change to be a problem.

At first no one in the organization had a strong vision for the end process. From level 2 onwards a gradually stronger and shared vision came in place. Consultants from outside and a management seminar played key roles in establishing the vision.

7 Conclusion

This study produced two main insights. First, as the organization matures, so does its capability to change. Moving from level 1-to-3 was a slow and sometimes frustrating process for Systematic, whereas the last two levels were reached in shorter time and with less frustration.

Second, the nature of the challenges facing Systematic's SPI project changed with increasing maturity. At the lower levels, gaining acceptance of and general commitment towards SPI constituted a major challenge. At the higher levels the problems changed towards understanding the requirements of the level (as described by CMMI - cf. Chrissis et al. 2003) and defining appropriate processes for them.

The first of these observations corresponds to results from other studies that show how the time required to reach the next maturity level decreases with higher maturity (cf. SEMA-reports at www.sei.cmu.edu). However, to our knowledge, no other study has documented the nature and challenges of SPI through maturity levels 1-to-5.

This chapter contributes to both SPI research and practice. Towards research, we contribute an understanding of the progress of processes, challenges, and benefits in long time SPI projects; Software developers and SPI practitioners will find much useful information in the description of Systematic's journey. Management may find the results and effects that Systematic realized most absorbing.

References

Ahern, Dennis M. Aaron Clouse and R. Turner (2001). *CMMI Distilled: A practical introduction to integrated process improvement*. SEI Series in Software Engineering, Addison-Wesley, Boston, MA, USA.

Borum, F. (1995). *Strategier for organisationsændring* (Strategies for Organizational Change). Handelshøjskolens Forlag, Copenhagen.

Burnes, B. (1996). *Managing Change*. 2nd Edition. Financial Times, Pitman Publishing.

Burnstein, I, Ariya Homyen, T. Suwanassart, G. Saxena and R. Grom (2002). A testing Maturity Model for Software Test Process Assessment and Improvement. Chapter 6.2 in Ted Daughtrey (Ed.) (2002). *Fundamental Concepts for the Software Quality Engineer.* American Society for Quality, Quality Press, Milwaukee, Wisconsin.

Chrissis, M. B., M. Konrad and S. Shrum (2003). *CMMI - Guidelines for Process Integration and Product Improvement.* SEI Series in Software Engineering, Addison-Wesley, Boston, MA, USA.

CMMI Product Team (2002). *Capability Maturity Model Integration (CMMI)*, version 1.1. CMMI for Software Engineering. Staged Representation. Software Engineering Institute: Pittsburgh, PA. p. 639.

Cooper, J. and M. Fisher (2002). *Software Capability Maturity Model (SA-CMM)* version 1.03. Pittsburgh, CarnegieMellon. Software Engineering Institute.

Deming, W. E. (1982) *Out of the Crisis*, MIT Center for Advanced Engineering Study, Cambridge, Mass.

Diaz, M. and J. King (2002). *How CMM Impacts Quality, Productivity, Rework, and the Bottom Line*. CROSSTALK The Journal of Defense Software Engineering, March 2002.

Goldenson, D. R. and J. D.Hersleb (1995). After the Appraisal: A systematic survey of process improvement, its benefits, and factors that influence success. Technical Report CMU/SEI-95-TR-009. Software Engineering Institute, Carnegie Mellon University, Pittsburgh

Handy, C. (2005). *Understanding Organizations*. 4th Edition. Penguin Global.

Hendry C. (1996). Understanding and Creating Whole Organizational Change Through Learning Theory. Human Relations, 49: 621- 41.

Kaplan, R. and D. Norton (1992). The Balanced Scorecard: Measures that drive performance. *Harvard Business Review*, (January-February): 71-79.

Kotter, J. P. (1996). *Leading Change*. Harvard Business School Press, Boston.

Krasner, H. and D. Houston (1998). Using the Cost of Quality Approach for Software. *CROSSTALK The Journal of Defense Software Engineering*, 11(11): 6-11.

Krasner, H. (2001). Accumulating the Body of Evidence for the Payoff of Software process Improvement. In: Software Process Improvement, IEEE Compouter Society press, pp. 519-539.

Kuvaja, P., J. Similä, L. Krzanik, A. Bicego, S. Saukkonen, G. Koch (1994). *Software Process Assessment and Improvement. The BOOTSTRAP Approach*. Blackwell Publishers.

Lewin K. (1951). *Field Theory in Social Science*. New York: Harper and Row.

Levine, L., R. Baskerville, J. L.L.. Link, J. Pries-Heje, B. Ramesh and S. Slaughter (2002). Discovery Colloquium: Quality Software Development @ Internet Speed, CMU/SEI-2002-TR-020, ESC-TR-2002-020, Carnegie Mellon Software Engineering Institute, Pittsburgh.

Mathiassen, L., J. Pries-Heje and O. Ngwenyama (eds.) (2001). *Improving Software Organizations: From principles to practice*. Addison-Wesley.

Paulk, M. C., C. Weber, B. Curtis and M. B. Chrissis (1995). *The Capability Maturity Model: Guidelines for improving the software process*, Addison-Wesley, Reading.

Schillings, M.L., W. Hoefsloot, D.F. Stegeman and M.J. Zwarts (2003). Relative Contributions of Central and Peripheral Factors to Fatigue During a Maximal Sustained Effort. *Journal of Applied Physiology*, 90: 562-568.

Stake, R. E. (2000). Case Studies, In: *Handbook of Qualitative Research*, N. K. Denzin and Y. S. Lincoln, editors, SAGE, Thousand Oaks.

Steinmark, A. R. (2004). Moral og modenhed. Leder i Computerworld, Danmarks IT-avis, 11. juni 2004.

Taylor, F. W. (1991). *The Principles of Scientific Management.* New York: Harper Bros., 1911

Weick, K. E. and R. E.Quinn (1999). Organizational Change and Development. *Annual Review of Psychology,* 50: 361-386.

Aaen, I., J. Arent, L. Mathiassen and O. Ngwenyama (2001). A Conceptual MAP of Software Process Improvement, *Scandinavian Journal of Information Systems,* 13(1), 2001.