Enabling Software Process Improvement:
An Investigation of the Importance of Organizational Issues

by

Tore Dybå

Doctoral Dissertation

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

Doctor Ingeniør

Department of Computer and Information Science
Faculty of Physics, Informatics and Mathematics
Norwegian University of Science and Technology

Trondheim, November 5, 2001

ISSN 0802-6394
ISBN 82-471-5371-8

For questions about this publication, please contact:

Tore Dybå
SINTEF Telecom and Informatics
N-7465 Trondheim
Norway

tore.dyba@sintef.no

IDI Report 7/01
NTNU 2001:101

Printed in Norway by
Tapir Trykkeri, Trondheim, November 2001.

To the Memory of My Father

Per Torbjørn Dybå

9th April 1936 – 12th September 2000

Abstract

Software development and maintenance involves organizational as well as technical issues. While software engineering has been offered as a way of resolving the intrinsic technical problems, the organizational problems need another approach. Still, the typical response to the persistent problems in software development has been to apply even more rigorously the principles of engineering. Organizational issues are often ignored or not properly addressed in much of the recent literature, with only a handful of studies being reported.

To help fill this gap, this doctoral study was initiated to explore the relative importance of organizational issues in software process improvement. The position taken is strongly influenced by socio-technical theory with its central conception that organizations are both social and technical systems, and that the core of the software organization is represented through the interface between the technical and human (social) system. The fundamental assumption is that process improvement is a socially constructed learning process and that a commitment to learning rather than to "best practice" models is needed to accomplish improvements in software development.

The research presented in this dissertation is based on several empirical studies performed within the context of the European Systems and Software Initiative and the Norwegian software process improvement projects SPIQ and PROFIT.

The overall research design for the study was a sequential, mixed method design consisting of two major phases: first, a qualitative model-building phase, then a quantitative model-testing phase. The model-building phase was grounded in prior consulting experience in software development, in observations done in a pilot case study as well as in a multiple case study of twelve organizations, and in an extensive literature review. The model-testing phase focused on testing the associations between the key factors for success and the outcome variable by a quantitative survey among 120 software organizations.

The findings from the investigations suggest that the key to successful learning is a continuous and simultaneous dialectic interplay between the knowledge that the organization has established over time, and the knowing of the organization's members in their respective contexts. Also, the findings indicate that success depends critically on six organizational factors. Finally, the findings show that there are important differences between small and large software organizations, specifically in the ways in which they react to unstable and changing stimulus situations.

The conclusion on the research problem is that process improvement cannot be managed, but only enabled through the space in which the software organization creates the possibilities for sensemaking, knowledge creation, and purposeful action.

The main contribution of the dissertation is to increase the understanding of the influence of organizational issues by empirically showing that they are at least as important as technology for succeeding with software process improvement. This suggests that software organizations that want to prosper in the 21st century should synergistically combine technology with social collaboration to become learning software organizations.

Acknowledgements

I am in great debt to several persons who have provided help, support, and encouragement throughout my work with the doctoral studies during the last four years. First of all, I would like to thank my advisors Reidar Conradi and Tor Stålhane for their extensive comments, stimulating discussions, and long-lasting cooperation. I would also like to thank you for not giving up on me during my periods of drifting between software engineering and organization development.

SINTEF Telecom and Informatics has remained a firm foundation and an encouraging employer. Thanks to Aage Thunem, Eldfrid Øvsti Øvstedal, and Mette Vestli for giving me this opportunity. I also acknowledge the Research Council of Norway for their financial support of my work through grant no. 118206/221.

I would like to thank Reidar Palmstrøm and Hans Erik Stokke (now Protek Telsoft AS) for their kind support and cooperation during the pilot study at Nera AS. I also acknowledge and thank all informants of the case studies and all respondents of the survey investigations for their willingness to participate in my inquiries.

Furthermore, I acknowledge and thank all my research colleagues and industrial partners in the SPIQ and PROFIT programs for their cooperation and useful discussions. A special thanks goes to Nils Brede Moe, Geir Kjetil Hanssen, and Kari Juul Wedde (now Clustra AS) at SINTEF, Torgeir Dingsøyr and Maria Letizia Jaccheri at NTNU, Dag Sjøberg and Magne Jørgensen at the Simula Research Laboratory, and Tor Ulsund at Bravida Geomatikk.

I would also like to acknowledge my family. A specific thanks goes to my mother and father for supporting and believing in me during all years. Sadly, my father died of cancer before this work was finished. It was hard to see him struggle with such bravery during these years – and lose. I miss you.

Finally, I would like to thank Trude and our children Sondre, Erlend, and Vilde for keeping up with me during these important years of our lives. I put a heavy burden on you all, but most of all on Trude. I am deeply grateful for your enduring support and understanding for why I couldn't always participate in family activities, social gatherings, and vacations. Thank you.

NTNU, November 5, 2001

Tore Dybå

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

"The scientist is not a person who gives the right answers, he is one who asks the right questions."

– Claude Levi-Strauss

In this chapter, the background to the research is explained by focusing on the persistent problems in software development and the importance of organizational issues in improving the software process. Furthermore, the chapter describes the research context, the research problem and questions, the claimed contributions, and the research method. Finally, the outline of the thesis is described.

1.1 Background to the Research

Software is the new infrastructure of the 21st century. It pervades our society, and is becoming increasingly critical for businesses as well as for leisure time and entertainment. Software is being used in more and more sensitive applications, ranging from nuclear reactors, airplanes, and air-traffic control to telecommunication networks, banking, and automobiles. It is fundamental to economic success, scientific and technical research, and national security. At the same time, there is a growing perception that software quality is as vulnerable as Achilles' heel in developing high-quality products. The chronic "software crisis", that software is (almost) always later than expected, more expensive than planned, and with less or even wrong functionality, has become even more critical as software pervades our day-to-day lives.

Despite impressive technological advances, the exponentially increasing demands, complexity of the problems, and scale of work addressed by software, seem to outpace our ability to develop and maintain software. Thus, in addition to being used in increasingly more sensitive applications, software is also becoming larger and more complex. Larger applications, with a greater volume of code means increased risk of error unless we dramatically improve the error rates of software. At the same time, the increasing complexity of industrial software systems make them progressively more difficult to test. In combination, these trends expose us to even greater risks of software failures and corresponding disasters as we use software in increasingly critical applications.

1.1.1 *Problems in software development*

Since the term "software engineering" was first coined at the historic NATO Conference in Garmisch-Partenkirchen in 1968 (Naur and Randell, 1969), there has been a widespread consensus that there are problems with the development and maintenance of software. These problems were later discussed by Brooks (1975, 1995) in his classic book *The Mythical Man-Month*, and the term "software crisis" was coined to embrace the problems in software development – they were seen as a werewolf that needed slaying with a "silver bullet" (Brooks, 1987).

Following Aristotle, Brooks (1987) distinguished between the essential and the accidental problems in software development. The *essential* problems are the difficulties inherent in the nature of software, which we have to learn to live with as long as we continue to develop software. The *accidental* problems, on the other hand, are those difficulties that attend software development because of the historical progress of our practice. They are not inherent and can be removed if we can identify and understand their causes.

More recently, Gibbs (1994) wrote a widely cited article in *Scientific American* entitled *Software's Chronic Crisis*, focusing on a wide range of problems in software development:

> Studies have shown that for every six new large-scale software systems that are put into operation, two others are cancelled. The average software development project over-shoots its schedule by half; larger projects do worse. And some three quarters of all large systems are 'operating failures' that either do not function as intended or are not used at all (*ibid*., pp. 72-73).

Also, the PITAC report (Joy and Kennedy, 1999) singled out the problems with software quality and software development as a key obstacle for the way we live, learn, work, and play in the 21st century. Similar to the above passage from Gibbs (1994), the PITAC report quotes figures from the Standish Group stating that 73 percent of software projects are late, substantially over budget, canceled, or outright failures.

The term "software engineering" was coined at the NATO conference as an answer to the problems in software development. It was deliberately provocative, implying the need for software development to be based on the principles and practices seen in engineering. Thus, the point of departure for most of the subsequent efforts in addressing the problems in software development has been to treat the entire task of software development as a process that can be improved through engineering methods.

Hoare (1984), for example, considered the "rise of engineering" and the use of "mathematical proof" in software development as a promise to "transform the arcane and error-prone craft of computer programming to meet the highest standards of a modern engineering profession." (*ibid*., p. 8). Likewise, Lehman (1989) focused on reducing uncertainty in the development process through the "engineering of software", while Basili (1996) put the case for a scientific approach to software development, stating that "*Software engineering is a laboratory science*" (*ibid*., p. 443, emphasis in original).

Humphrey (1989) recognized that the key problems in software development are not technological, but managerial in nature. Consequently, he developed a framework for managing and improving the software process, which was later known as "The Capability Maturity Model for Software" (CMM) (Paulk *et al.*, 1995). This framework reflects the underlying premise of software process improvement (SPI), that "the quality of a software product is largely governed by the quality of the process used to develop and maintain it." (*ibid.*, p. 8). Consistent with the views of software engineering, the CMM is rooted in the engineering tradition, emphasizing predictability and improvement through the use of statistical process control. Humphrey (1989) formulated his fundamental view in this way: "If the process is not under statistical control, sustained progress is not possible until it is" (*ibid.*, p. 3).

As these examples, from some of the most influential academic leaders within the software community, show, software development and software process improvement are strongly rooted in the rationalistic traditions of engineering. Indeed, most of the writings to date can be seen to have antecedents in the industrial models devised by Frederic Winslow Taylor and Henry Ford, and also in the notion of bureaucracy described by Max Weber. As a result, the move toward improvements in software development processes has remained technology driven.

However, software development and software process improvement involves organizational as well as technical issues. While software engineering has been offered as a way of resolving the intrinsic technical problems, the organizational problems need another approach. However, *both classes of problems need to be jointly resolved to improve the process of software development* – i.e. they are co-producers of the outcome. It is, therefore, my contention that we cannot expect to succeed with the practice of SPI if we continue to approach SPI research with one eye closed.

1.1.2 The importance of organizational issues in SPI

Concern with the relationship between the problems in software development and organizational issues is not a new phenomenon. More than 25 years ago, Lucas (1975) wrote a classic book on why information systems fail, suggesting that "the primary cause for system failure has been organizational behavior problems." (*ibid.*, p. 3). There is, however, an increasing amount of evidence to suggest that organizational issues are now more important, and will become even more critical, to the successful improvement of software development than they were when Lucas (1975) wrote his book (Bennetts *et al.*, 1999; Bjerknes, Dahlbom *et al.*, 1990; Dahlbom and Mathiassen, 1993, 1997; Doherty and King, 1998a; Kawalek and Wastell, 1999). This was also the focus of the October 1993 issue of *Communications of the ACM* and the May/June 2001 issue of *IEEE Software*, which had a focus on organizational change.

Despite this increasingly recognized importance of organizational issues within the SPI community, the typical response to the persistent problems in software development has been to apply even more rigorously the principles of engineering (e.g. Florac and Carleton, 1999). Organizational issues are often ignored or not properly addressed in much of the recent literature, with only a handful of studies being reported (e.g. El Emam *et al.*, 1999, 2001; Goldenson and Herbsleb, 1995; Guinan *et al.*, 1998; Sawyer and Guinan, 1998). Furthermore, organizational issues are only treated implicitly, or in many cases not at all, even by managers who perceive organizational issues to be of more importance than technical issues in determining the successful outcome of systems development projects (Doherty and King, 1998b). There is, therefore, a pressing need for empirical research to investigate the importance of organizational issues in SPI.

The position taken in this thesis regarding organizational issues is strongly influenced by socio-technical theory (Trist and Bamforth, 1951; Trist 1981). Its central conception is that organizations are both social and technical systems, and that the core of the software organization is represented through the interface between the technical and human (social) system. Thus, I define organizational issues within the context of SPI as *any distinct area on the interface between the technical system and the social system, which can enable software process improvements*. This definition encompasses a wide range of organizational issues that need to be addressed in SPI, which others have classified as managerial (Humphrey, 1989, 1997), behavioral (Curtis *et al.*, 1988), human (Wastell, 1999), or people (DeMarco and Lister, 1999) issues.

From a software engineering perspective, the world is composed of problems whose existence is distinct from the methods, tools, and techniques of software engineering. The technical rationality behind this worldview emphasizes "objective truths" and global "best practices", at the expense of local context and expertise. An important aspect of socio-technical theory, however, is the belief that there may be *many optimal solutions* – or best ways – to a specific problem, since the "joint optimization" of a particular technical and human system can be implemented in several ways that can be equally efficient. This position is particularly relevant for SPI because, as I already have argued, of the intimate interdependence between technical and organizational issues.

In addition to "joint optimization", socio-technical theory also emphasizes the principles of "minimum critical specification", "autonomous work groups", and "co-determination" (Trist, 1981). Instead of a search for global best practices, therefore, the thesis points to the importance of context-sensitive learning (Walz *et al.*, 1993) and to follow the rules suggested by the situation at hand (Robinson *et al.*, 1998). Furthermore, I envisage software development largely as a human based intellectual activity (Weinberg, 1971, 1998) that involves constant negotiation and renegotiation among and between the social groups shaping the software.

Thus, the fundamental assumption behind the investigation presented in this thesis is that *software process improvement is a socially constructed learning process* and, consequently, that a commitment to learning rather than to "best practice" models is needed to accomplish improvements in software development. Therefore, the thesis focuses on studying these learning processes and the key factors that enable improvements in organizational performance.

1.2 Research Context

The research presented in this thesis has, for the most part, been performed within the context of the following three projects:

TELMET (Telecommunications Metrics Approach) was a Process Improvement Experiment (PIE) within Nera AS supported by the European Systems and Software Initiative (ESSI) that lasted from April 1996 to April 1998. The overall goal of TELMET was to improve the quality and productivity of telecommunication software systems development by introducing metrics in the testing and inspection processes. In addition, the project acted as the first step in Nera's effort to advance from qualitative to quantitative goal setting. The technical objectives were: (1) to identify and define software metrics to improve product quality; (2) to analyze the effect of quantitative goal setting and measurement; and (3) to validate the chosen metrics in order to implement post PIE actions.

I was part of the PIE team, with a general responsibility for training and methodology support during all phases of the measurement program. Specifically, I had responsibility for developing measurement plans according to **ami** and GQM (see *Chapter 2*) and performing statistical data analysis as data became available (see Dybå *et al.*, 1997; Stålhane *et al.*, 1997b).

SPIQ (Software Process Improvement for better Quality) was a large National SPI program, partly funded by the Research Council of Norway, which lasted for three years from January 1997 to December 1999. The objective of SPIQ was to increase the competitiveness and profitability of Norwegian IT-industry through a systematic and continuous approach to process improvement. The goal of SPIQ was twofold: (1) to establish an environment for process improvement in the software organizations associated with SPIQ, and (2) to transfer and diffuse the knowledge gained to the remaining IT-industry in Norway through training, seminars, and conferences.

The twelve software companies that actively participated in SPIQ covered organizations of different sizes with a wide range of products and markets, developing either software or combined software and hardware products. Some companies belonged to the traditional electronics based IT-industry, while others belonged to the new dot-com industry. Besides SINTEF, the Norwegian University of Science and Technology (NTNU), and the University of Oslo made up the research partners in the program.

I was one of three Ph.D. fellows in SPIQ. My main role was to assist the member organizations with measurement based improvement of their software development processes relative to company-specific quality goals (see Conradi and Dybå, 2001; Dybå, 2000c, d; Dybå and Moe, 1999; Stålhane *et al.*, 1998). In addition, I also had the editorial responsibility for the SPIQ Methodology Handbook (see Dybå, 2000e).

PROFIT (Process improvement for IT industry) is a National SPI program partly funded by the Research Council of Norway, which started in March 2000, and will continue throughout 2002. The overall goal of PROFIT is to "increase the competitiveness and profitability of the Norwegian IT-industry through focusing on learning and process improvement in the continuously changing environment of the software business". Like SPIQ, PROFIT involves twelve software companies, with SINTEF, NTNU, and the University of Oslo as the research partners.

Focus areas for PROFIT are software process improvement under uncertainty and change. PROFIT tries to deal with these issues by focusing on helping small and medium-sized software-intensive companies create learning software organizations. As in SPIQ, the software companies participating in PROFIT run improvement projects according to the ESSI model, with an improvement project as a layer on top of a real development project.

As in TELMET and SPIQ, my role in PROFIT is to help software organizations in improving their software development processes relative to company-specific quality goals while at the same time contributing to the body of knowledge within SPI (see Dybå, 2000a, b).

Additionally, my personal experience from 15 years of software development and consulting for a wide range of customer and supplier organizations in industry and government has strongly influenced both the content and the approach of the present investigation.

1.3 Research Problem and Questions

In this section, the research problem and questions are explained. Although it has partially been covered by the discussion in *Section 1.1*, I comment on the title of the thesis first to reveal some of the most important assumptions.

Thesis Title:

Enabling Software Process Improvement:
An Investigation of the Importance of Organizational Issues

The title of the thesis has four main parts, which are informally explained below:

- **Enabling**: The perspective taken in this thesis is a recognition of the need to enable software process improvement rather than trying to control it. Therefore, the focus is on enabling SPI rather than on efforts to manage it.
- **Software Process Improvement**: The fundamental assumption of SPI taken in this thesis is that changes to the process cause changes in the outcome. Thus, rather than being interested in the process itself, the thesis is more concerned about the process outcomes.
- **Investigation**: The thesis includes a substantial theoretical investigation with an extensive literature review of more than 600 references. The empirical investigation includes 13 qualitative case studies as well as the main, quantitative survey of 120 software organizations.
- **Importance of Organizational Issues**: The aim of the thesis is to investigate organizational issues that have a substantial importance in enabling SPI. The thesis focuses on two classes of such issues: (1) the socially constructed learning processes, and (2) the key factors that facilitate improvements in organizational performance.

Despite the increasingly recognized importance of organizational issues within the SPI community, few empirical studies have been reported. To help fill this gap, the doctoral study, reported in this thesis, was initiated to explore the relative importance of organizational issues by investigating the learning processes and key factors for success in SPI.

Thus, the problem addressed in this thesis is:

Research Problem:

How can software organizations enable software process improvement?

Essentially, I argue that SPI cannot be managed, but only enabled through the space in which the software organization creates the possibilities for sensemaking, knowledge creation, and purposeful action.

To narrow the focus of the investigation, the research problem addressed by this thesis can be summarized by the following three questions:

Research Questions:

Q1: What are the key learning processes in successful software organizations?

The first question focuses on identifying the key processes that are part of a successful software organization's learning cycle. A theoretical investigation and a set of qualitative case studies answer the question.

Q2: What are the key factors of success in software process improvement?

The second question focuses on identifying the key factors for success in SPI. A theoretical investigation and a quantitative survey answer the question.

Q3: What are the relationships between organizational context and modes of learning in software organizations?

The third question focuses on finding the relationships between the two major modes of learning, exploitation and exploration, and the software organizations' context. The question is answered by the quantitative survey.

Taken together, the answers to these questions constitute the main contributions of this thesis. These contributions are put forward in the next section.

1.4 Claimed Contributions

The objective of the thesis is to investigate the importance of organizational issues in SPI. Based on this objective and the answers provided to the research questions posed in the previous section, I claim that the thesis contributes with unique theoretical, methodological, and practical knowledge. The main contributions are:

- **An increased awareness of the importance of organizational issues in SPI**. For the most part, SPI has been preoccupied with technical issues at the expense of organizational issues. The overall contribution of the thesis, therefore, is to empirically show that organizational issues are, indeed, important for succeeding with SPI.
- **A dynamic model of SPI**. Based on my personal experience and observations, and the extensive literature review, I have developed a dynamic model of SPI. The model is inspired by social constructivism and open systems theory, and it consists of the following four major elements: organizational context, learning cycle, facilitating factors, and organizational outcome. The model is presented in *Chapter 5* and answers research question *Q1*.
- **Key factors for success in SPI**. As an important part of the dynamic model of SPI, I defined a set of key factors for success. The factors are theoretically justified and defined in *Chapter 5*, empirically justified in *Chapter 7*, and validated in *Chapter 8*, providing the answer to research question *Q2*.
- **An improvisational approach to SPI**. The results of my investigations showed that there are important differences in the way large successful and small successful organizations react to environmental conditions and how they deal with SPI. Consequently, I propose an approach to SPI in small organizations based on improvisation. The approach is described in *Chapter 9*, and provides the answer to research question *Q3*.

Moreover, several papers have been published during the period of the doctoral studies with additional contributions to the national and international body of knowledge in SPI:

Journal articles:

- Dybå, T. (2000a) An Instrument for Measuring the Key Factors of Success in Software Process Improvement, *Empirical Software Engineering*, Vol. 5, No. 4, December, pp. 357-390.
- Dybå, T. (2000b) Improvisation in Small Software Organizations, *IEEE Software*, Vol. 17, No. 5, September-October, pp. 82-87.
- Dybå, T. (1999) Assessment-based Software Process Improvement, *Telektronikk*, Vol. 95, No. 1, pp. 37-47.
- Dybå, T. and Skogstad, Ø. (1997) Measurement-Based Software Process Improvement, *Telektronikk – Special Issue on Software Quality*, Vol. 93, No. 1, pp. 73-82.

Books and book chapters:

- Dybå, T. (Ed.) (2000e) *SPIQ – Software Process Improvement for better Quality: Methodology Handbook* (in Norwegian), IDI Report 2/2000, Trondheim, Norway:

Norwegian University of Science and Technology (NTNU), Faculty of Physics, Informatics and Mathematics.

- Dybå, T. (2000c) Planning Organizational Improvement Programs, in T. Dybå (Ed.), *SPIQ – Software Process Improvement for better Quality: Methodology Handbook* (in Norwegian), IDI Report 2/2000, Trondheim, Norway: NTNU.
- Dybå, T. (2000d) Planning Process Improvement Experiments, in T. Dybå (Ed.), *SPIQ – Software Process Improvement for better Quality: Methodology Handbook* (in Norwegian), IDI Report 2/2000, Trondheim, Norway: NTNU.

Conference papers:

- Conradi, R. and Dybå, T. (2001) An Empirical Study on the Utility of Formal Routines to Transfer Knowledge and Experience, *Proceedings of the 8th European Software Engineering Conference (ESEC'2001)*, Vienna, Austria, 10-14 September.
- Dybå, T. and Moe, N.B. (1999) Rethinking the Concept of Software Process Assessment, *Proceedings of the European Software Process Improvement Conference (EuroSPI'99)*, Pori, Finland, 25-27 October.
- Stålhane, T., Wedde, K.J., and Dybå, T. (1998) Data Driven Improvement for SMEs, *Proceedings of the European Software Process Improvement Conference (EuroSPI'98)*, Gothenburg, Sweden, November 16-18.
- Dybå, T., Stålhane, T., and Palmstrøm R. (1997) Experience of Goal-Oriented Measurement using **ami** and GQM, *Proceedings of the 8th European Software Control and Metrics Conference (ESCOM'97)*, Berlin, Germany, 26-28 May.
- Stålhane, T., Dybå, T., and Palmstrøm, R. (1997) Experience of Introducing Goal-Oriented Measurement, *Proceedings of the 8th International Workshop on Software Technology and Engineering Practice (STEP'97)*, London, July 14-18.

1.5 Research Method

The overall research design for the study is shown in *Figure 1.1*. It is a sequential, mixed method design consisting of two main phases: first, a qualitative model-building phase, then the main, quantitative model-testing phase. The *model-building* phase was grounded in my prior, personal experience as a consultant, in my observations in the case studies, and in the extensive literature review. The *model-testing* phase focused on testing the associations between the key factors for success and the outcome variable, SPI success, by a quantitative survey.

Together, these two phases combine "process research" with "factor research" (Newman and Robey, 1992), which can be seen as two major approaches to SPI research with antecedents in MIS (Management Information System) research. *Process research* is typically focused on the dynamics of individual projects using qualitative methods such as interviews and participant-observations. *Factor research*, on the other hand, is typically focused on assessing the relationships between key causal and outcome variables by collecting survey data from a large number of organizations.

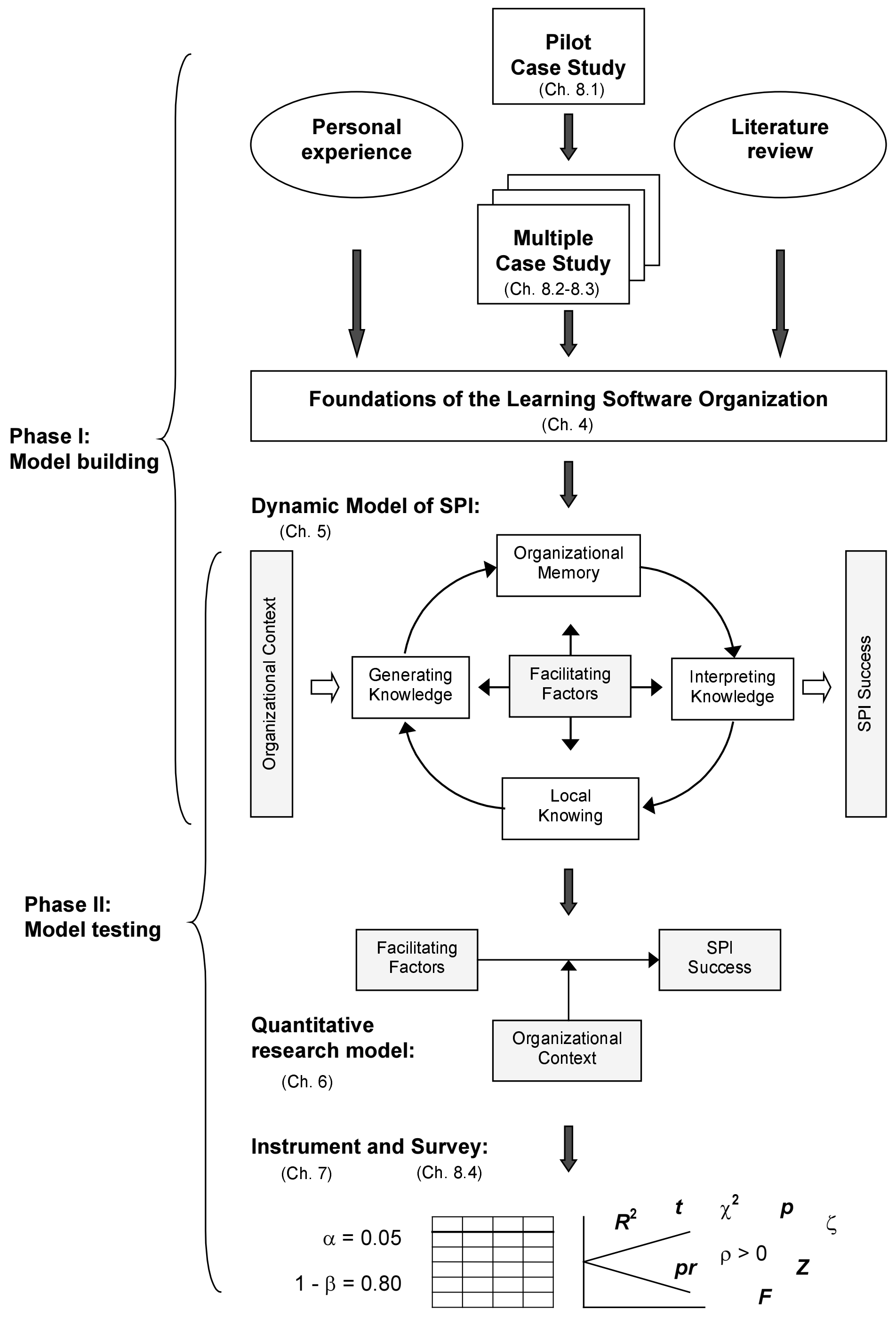


Figure 1.1: Overall research design consisting of a qualitative model-building phase followed by a quantitative model-testing phase.

The unit of analysis, from which original data observations were obtained, was the software organization, which was defined as *a whole company or an independent business unit inside a larger company that has software development as its primary business*.

The process research approach in the model-building phase included a single, exploratory, in-depth pilot case study (*Case study A*: TELMET, *Section 8.1*), followed by a more explanatory, cross-case analysis of twelve software organizations (*Case studies A* and *B*: SPIQ, *Sections 8.2- 8.3*; and Dybå, 2000e). Collectively, these case studies had a considerable influence on the model ultimately measured and tested by the following survey. Together with the literature review and my prior experience as a consultant, the experience gained and data collected during this phase of the research provided the *foundations* (*Chapter 4*) for developing the *dynamic model of SPI*, which is described in *Chapter 5*.

The hypotheses regarding the key factors of success in SPI developed in the model-building part of the study were tested using an explanatory, cross-sectional, field survey design (see *Section 8.4*). It involved 120 software organizations, investigating to what extent the identified practices were implemented and what impact they had on SPI success. In this main part of the study, I used a mailed questionnaire as the data collection instrument (see *Appendix F*). The instrument was developed specifically for this study and results of reliability and validity analyses showed that the instrument has desirable psychometric properties (see *Chapter 7*).

The primary intent of the survey was to test the existence of statistical associations among the identified factors and SPI success, thus providing quantitative evidence for the importance of organizational issues in SPI, and the generalizability of results.

In addition to the extensive literature review, my personal experience, and general observations in TELMET, SPIQ, and PROFIT, *the investigation included eight empirical studies*:

(1) A pilot case study on goal-oriented measurement (*Case A*, *Section 8.1*).

(2) A multiple case study on participative software process assessment (*Case B*, *Section 8.2*).

(3) A multiple case study on the utility of formal routines to transfer knowledge and experience (*Case C*, *Section 8.3*).

(4) An exploratory study on SPI success factors (*Case D*, *Section 7.2*).

(5) An expert review of proposed factors (*Section 7.2*).

(6) The development of a measurement instrument (*Sections 7.3-7.4; 7.6-7.9*).

(7) A pilot test of the measurement scales and the overall instrument (*Section 7.5*).

(8) The main quantitative survey (*Section 8.4*).

1.6 Outline of the Thesis

The thesis is organized in five parts, which reflects the phases of the research outlined in the previous section.

Part I, **State of the Practice and State of the Art**, contains two chapters related to the theoretical investigation of the current practices and research in SPI:

- *Chapter 2: Approaches to Software Process Improvement*. This chapter surveys the major approaches taken to SPI. Model-based approaches are compared with analytical approaches, key factors for success are reviewed, relevant terms are defined, and the relationships between SPI and organizational outcome are examined.
- *Chapter 3: Approaches to Empirical Software Engineering Research*. This chapter surveys and compares the major research approaches in empirical software engineering. In addition, the chapter presents an overview of some of the most important research initiatives in SPI.

Part II, **Theoretical Framework**, contains two chapters about the model resulting from the model building phase of the investigation: the theoretical foundations and the overall model including a number of hypotheses regarding the key factors for success in SPI:

- *Chapter 4: Foundations of the Learning Software Organization*. This chapter describes the underlying assumptions and foundations of the learning software organization, which form the basis upon which the dynamic model of SPI is constructed. The following foundations are described: *Social learning*, which is focused on SPI as a social, collaborative activity within the context of a learning software organization. *Sensemaking*, which is aimed at constructing meaning and expressing the basic assumptions and values that are vital to the software organization and its members. *Knowledge creation*, which is aimed at generating new knowledge and new competencies that enable or broaden the software organization's potential range of actions. *Purposeful action*, which is aimed at using the new interpretations and new knowledge to construct improved courses of action.
- *Chapter 5: A Dynamic Model of Software Process Improvement*. This chapter proposes a dynamic model of SPI for examining a software organization's learning capability and the processes that facilitate sensemaking, knowledge creation, and purposeful actions within the context of a learning software organization. Furthermore, the chapter describes the justification, conceptualization, and definition of each of the facilitating factors in the model and develops a number of hypotheses regarding their associations with SPI success. The model is grounded in my personal experience and observations in the case studies, related to the extensive literature review, and focused on the three research questions posed in *Section 1.3*.

Part III, **Empirical Investigation**, contains the details of the research methodology, the development of the measurement instrument used in the survey, the results of testing the hypotheses in the model, and a discussion of the results:

- *Chapter 6: Research Methodology*. This chapter describes the justification for and the assumptions behind the research approach used in this study, as well as the details of the methods used to develop the theory, operationalize the measures, collect the data, test the hypotheses, and answer the research questions.
- *Chapter 7: Measurement Instrument*. This chapter details the research method further and describes the design of the instrument used to measure the key factors of success in SPI. The chapter provides a synthesis of the prescriptions for success found from the literature review, which was also confirmed by empirical studies among both researchers and practitioners. The results of reliability and validity analyses show that the instrument has desirable psychometric properties.

- *Chapter 8: Empirical Results*. This chapter reports on the results of using the measurement instrument to test the hypotheses regarding the key factors for success in SPI and explore the relationships between the success factors and a set of contextual variables. Additionally, the chapter reports on three separate studies regarding (1) the experience with goal-oriented measurement, (2) the experience with participative software process assessment, and (3) the utility of formal routines to transfer knowledge and experience.
- *Chapter 9: Discussion and Implications*. This chapter contains the discussion of the results of the investigations in the thesis. The discussion is related to the learning processes in SPI, to the key factors of success, and to the effects of organizational context. Furthermore, the implications for theory and methodology are discussed using a dialectical approach. Finally, the implications for practice are discussed in terms of practical recommendations for successful SPI.

Part IV, **Conclusions**, summarizes the major findings of the thesis.

- *Chapter 10: Conclusions*. This chapter presents the conclusions on the research problem and questions and states the claimed contributions of the thesis. Furthermore, it includes an evaluation of the thesis with respect to both rigor and relevance. Finally, the limitations to the study are described along with recommendations for further research.

Part V, **Appendices**, contains additional material that has not been included in the other parts of the thesis.

- *Appendix A: Terminology*. This appendix provides a list of abbreviations and definitions of the most central terms used in the thesis.
- *Appendix B: Interview Guide*. This appendix contains the interview guide used in five of the qualitative case studies regarding the utility of formal routines to transfer knowledge and experience.
- *Appendix C: Example of Results from an Exploratory Study in CX*. This appendix includes the results from one of the four exploratory studies that were conducted to validate the literature-derived factors enabling SPI success.
- *Appendix D: Original List of Facilitating Factors and Indicators*. This appendix contains the original list of facilitating factors and indicators based on the literature review and the exploratory studies in the SPIQ companies.
- *Appendix E: Measurement Instrument*. This appendix includes the instrument developed to measure the key factors of success in software process improvement.
- *Appendix F: Questionnaire*. This appendix contains the survey questionnaire used in the quantitative part of the investigation based on the measurement instrument. Also, it contains questions regarding the characteristics of the respondent, and the respondent's company.
- *Appendix G: Survey Data*. This appendix contains the raw survey data collected from the 120 software organizations answering the questionnaire used in the quantitative survey.

Finally, the thesis includes a list of all references cited throughout the thesis. For stylistic reasons, I will use the term "we" instead of "I" in the rest of the thesis.

CHAPTER 2

Approaches to Software Process Improvement

"*Evolgato imperii arcano – for now had been divulged that secret of the empire, that emperors could be made elsewhere than at Rome.*"

– Publius Cornelius Tacitus

Understanding the "quality revolution" is an important prerequisite for understanding software process improvement (henceforth SPI). However, quality management is often obscured by both confusion and misunderstanding. This potential for misunderstanding is partly related to quasi-religious and sectarian controversies, and partly related to the most important feature of modern quality management: it directs attention to the improvement of production processes, and not simply to the characteristics of the product. In this respect, modern quality management opposes the original quality control principles of assuring that the characteristics of the end product fall within preassigned tolerance limits.

Along with traditional engineering, the quality management tradition focuses primarily on the *manufacturing* of material structures and their physical effects. However, software *development* is strikingly different, mainly concerned with flexible symbolic structures and their cognitive effects. In addition to computer instructions, software can also include a corresponding set of instructions for humans. To have a meaningful discussion of SPI we need to be clear about the characteristics of software, its usage, and the software development process (henceforth the software process).

In this chapter, therefore, we first describe the concepts related to the software process. Based on these concepts, we describe three intervention strategies or approaches to change the software process in order to bring about improvements in organizational performance. In *Figure 2.1*, the degrees of change to the *software process* are shown for each of these approaches.

First, *process assessment* is an intervention strategy that involves an appraisal or review of an organization's software processes without changing them. Second, *process improvement* is an intervention strategy that involves continuous or incremental change to the organization and its processes. Finally, *process innovation* is an intervention strategy that involves radical change to the organization with entire or partial replacement of its processes.

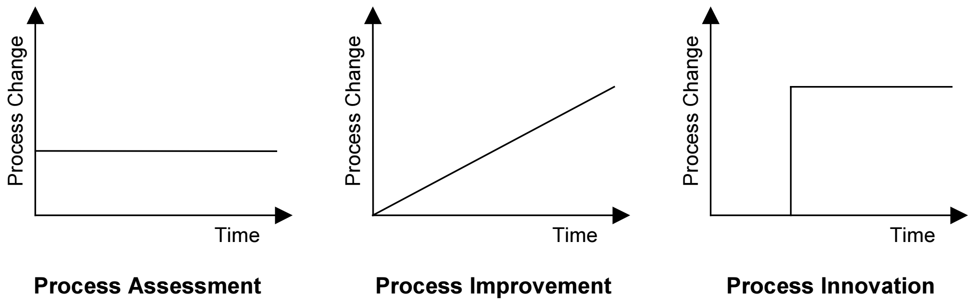


Figure 2.1: Degrees of process change for different intervention strategies.

2.1 *Software Process*

The software process defines the way in which software development and maintenance is organized, managed, measured, supported, and improved (Feiler and Humphrey, 1993; Montangero, 1999). Davenport (1993), for example, defined process as "a structured, measured set of activities designed to produce a specific output for a particular customer or market." (*ibid.*, p. 5), while Fuggetta (2000) defined a software process as "*a coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product*" (*ibid.*, p. 28, italics in original). Furthermore, Dowson (1993) noted that "All process work is ultimately directed at 'software process assessment and improvement.'" (*ibid.*, p. 57).

Consequently, software organizations have gradually come to realize that the key to successful delivery (with expected quality, on time, and on budget) lies in the effective management of their software processes. This is also reflected in modern quality management and in the underlying premise of software process management, that "the quality of a software product is largely governed by the quality of the process used to develop and maintain it" (Paulk *et al.*, 1995, p. 8). This relationship between process and outcome (e.g. time-to-market, cost, or customer satisfaction) can be explicitly expressed by the *causal relation*:

$$\textbf{\textit{Quality}}\ (\textit{Process}) \Rightarrow \textbf{\textit{Quality}}\ (\textit{Outcome}) \qquad (\textit{2.1})$$

However, the same change to a process does not always lead to a similar change in outcome. The relationship depends on the particular context (e.g. experience, organizational size, or environmental turbulence).

Figure 2.2 shows a path diagram with this relationship between process, context, and outcome. In general, we are not really interested in the process itself; rather, we are most concerned about the process outcomes. But, in order to achieve the desired outcomes, we have to implement the appropriate process (see *Section 4.4.1* and the path diagram in *Figure 4.6* for a general form of this theory of action).

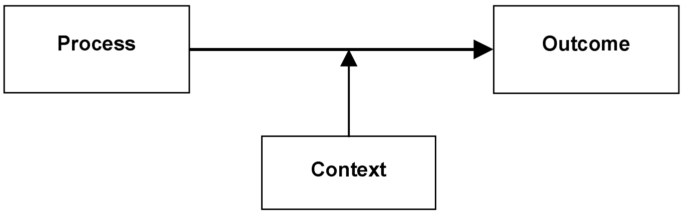


Figure 2.2: Relationship between process, context, and outcome.

2.1.1 *Process models*

Process models help us to become aware of and gain an increased understanding of the software process, they facilitate human understanding and communication, and provide process guidance (Conradi *et al.*, 1992; Curtis *et al.*, 1992; Madhavji, 1991). They focus on the stages, activities, and dynamics of software development by providing abstract descriptions of the tasks of software development, their interdependencies, and the resulting artifacts.

Overall, the value of a process model is in defining the phases and activities that lend structure to the apparent chaos that characterizes the software process (Parnas and Clements, 1986). By clarifying the activities, routines, and dynamic factors that determine the timing and trajectory of the development process, process models provide a framework by which software organizations can better manage the dynamic nature, and the introduction of new or improved methods, tools, and techniques. A limitation of these models, however, is that they hide important process details that are crucial for the success of software projects (Madhavji, 1991).

Software processes can be represented at various levels of granularity and precision (e.g. generic life-cycle models, organization specific models, project specific models), from several perspectives (e.g. functional, behavioral, organizational, or informational), and different degrees of scriptiveness (e.g. descriptive, prescriptive, or normative) (Derniame *et al.*, 1999; Finkelstein *et al.*, 1994). Also, they represent different philosophies toward practice, e.g. "do-it-right-the-first-time" vs. "fail-fast" (Sotirovski, 2001).

Traditional examples of process models include the *waterfall* model (Royce, 1970), the *iterative* enhancement model (Basili and Turner, 1975), the *transformational* model (Balzer, 1981), and the *spiral* model (Boehm, 1988). Comparisons of these models are provided in (e.g. Davis *et al.*, 1988). More recent examples of process models are the Rational Unified Process (Kruchten, 2000), eXtreme Programming (Beck, 1999), the Dynamic Systems Development Method (Stapleton, 1997), and Scrum (Schwaber and Beedle, forthcoming).

Also, the International Organization for Standardization (ISO) has developed a model for software life cycle processes (ISO/IEC 12207: 1995) and process assessment (ISO/IEC CD 15504: 2001). Moreover, the recent revision of the ISO 9000 family of standards encourages the adoption of a process approach to quality management and improvement, which is also reflected in the proposed revision of the guidelines for the application of ISO 9001:2000 to software (ISO/IEC WD 9000-3: 2001). Additionally, Bechtold (1996) provides a guidebook on how to improve the software process through process definition and modeling.

2.1.2 *Process infrastructure*

An appropriate process infrastructure in terms of resources and responsibilities is a necessary prerequisite for process management and improvement. Two types of infrastructures have been widely described in the literature: the *Experience Factory* (Basili and Rombach, 1991; Basili *et al*., 1994a) and the *Software Engineering Process Group* (Fowler and Rifkin, 1990). Furthermore, the Software Engineering Laboratory's SPI Guidebook (Jeletic *et al*., 1996), the Software Engineering Institute's IDEAL Handbook (McFeeley, 1996), the SPIQ Handbook (Dybå, 2000e), and the PROFES User Manual (PROFES, 1999) provide good descriptions of infrastructure for process improvement in general.

2.1.2.1 Software Engineering Process Group

The Software Engineering Process Group (SEPG) is meant to be a central force for process improvement within an organization. Its role is to maintain the overall view of current SPI efforts and to facilitate these efforts on a continuing basis by fostering collaboration among everyone in the organization who is involved with SPI.

The SEPG typically has the following ongoing activities (Fowler and Rifkin, 1990):

- Obtains and maintains the support of all levels of management.
- Facilitates software process assessments.
- Works with line managers, whose projects are affected by changes in software engineering practice, providing a broad perspective of the improvement effort and helping them set expectations.
- Maintains collaborative working relationships with software engineers, especially to obtain, plan for, and install new practices and technologies.
- Arranges for any training or continuing education related to process improvements.
- Tracks, monitors, and reports on the status of particular improvement efforts.
- Facilitates the creation and maintenance of process definitions, in collaboration with managers and engineering staff.
- Maintains a process database.
- Provides process consultation to development projects and management.

The SEPG is not part of product development but is, nevertheless, staffed by practitioners who have expertise in software engineering. It may also have, and at any rate should develop, expertise in process definition, organizational change, and technology related to improving or measuring quality.

2.1.2.2 Experience Factory

The Experience Factory (EF) is an organization that supports reuse of experience and collective learning within a software organization. The focus of the EF is on collecting data and lessons learned from projects and experiments, and then analyzing these data and packaging them into guide books, models, and training courses that can be spread to all areas of the development organization (Basili, 1989). The EF is different from the Project Organization, which focuses on the development and maintenance of applications. Their relationship is depicted in *Figure 2.3*.

In addition to the Software Engineering Laboratory (SEL), further examples of operational EFs include the Software Experience Center (SEC) setup by Fraunhofer IESE in Kaiserslautern and the Fraunhofer Center for Experimental Software Engineering in Maryland.

2.1.3 Process measurement

Evaluation of changes to the software process requires measurement. A number of practical guides for such measurement are available, e.g. the **ami** Handbook (Pulford *et al*., 1996), the GQM Method (Briand *et al*., 1996a; Gresse *et al*., 1995; van Solingen and Berghout, 1999), the Software Measurement Guidebook (Gaffney *et al*., 1995), the Goal-driven Measurement Guidebook (Park *et al*., 1996), the Guide to Practical Software Measurement (McGarry, 1998), and the Practical Software Measurement Guidebook (Florac *et al*., 1997).

Common to these guides is that they are based on goal-oriented measurement (Basili *et al*., 1994b), which helps ensure adequacy, consistency, and completeness of the measurement plan and the data collection procedures.

Goal-Question-Metric (GQM) is a pragmatic approach for goal-oriented measurement in software projects to select and implement relevant measures and indicators (Basili and Weiss, 1984; Basili and Rombach, 1988). It represents an approach for tailoring and integrating goals with models of the software processes, products, and quality perspectives of interest, based upon the specific needs of the project and the organization.

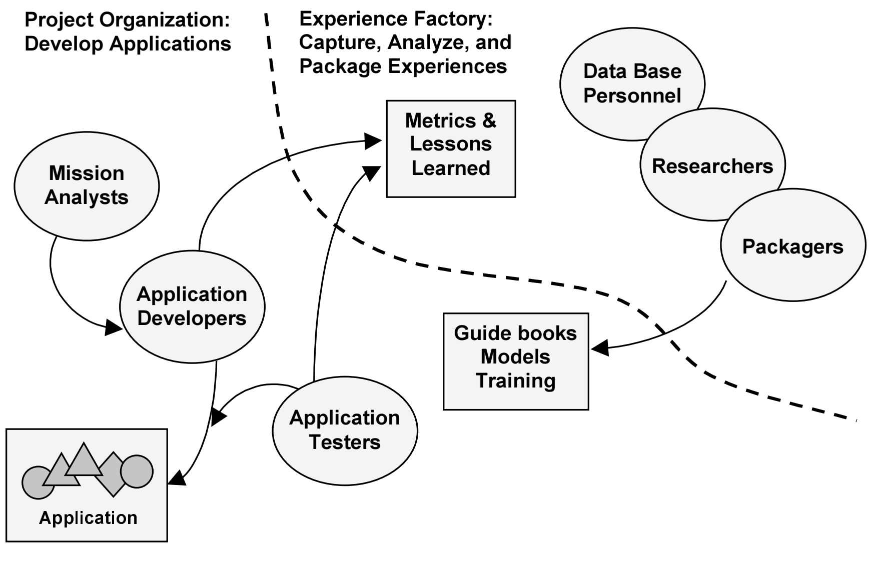

Project Organization: Develop Applications
Experience Factory: Capture, Analyze, and Package Experiences
Data Base Personnel
Metrics & Lessons Learned
Researchers
Mission Analysts
Packagers
Application Developers
Guide books Models Training
Application Testers
Application

Figure 2.3: The relationship between the Experience Factory and the Project Organization as implemented at NASA/GSFC (Basili *et al*., 1997, p. 284).

The GQM model has three levels (Basili *et al*., 1994b):

(1) **Conceptual level (goal)**: A goal is defined with respect to various models of quality, from various points of view, relative to a particular environment. Typical objects of measurement are *products*, *processes*, and *resources*.

(2) **Operational level (question)**: A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed, based on some characterizing model. Questions try to characterize the object of measurement with respect to a selected quality issue, and to determine its quality from the selected viewpoint.

(3) **Quantitative level (metric)**: A set of data is associated with every question in order to answer it in a quantitative way. The data can be either *objective* or *subjective*.

A GQM model is a hierarchical structure starting with a goal as shown in *Figure 2.4*. The goal is subsequently refined into a set of questions, and each question is then refined into metrics. The metrics reflect the actual data needed to answer the questions. The same metric can be used in order to answer different questions under the same goal.
As an aid in the process of defining the goals, questions, and metrics, the CEMP-project has provided a process model and a series of useful templates (Gresse *et al*., 1995).

A practical text on establishing a measurement program has been made by the Software Engineering Laboratory (Bassmann *et al*., 1995). Furthermore, the ISO Software Measurement Process Framework (ISO/IEC CD 15939:2001) defines a generic measurement process. Texts that present experience with the implementation of measurement in software organizations include (Fenton and Pfleeger, 1996; Grady, 1992; Grady and Caswell, 1987; Kitchenham, 1996; Möller and Paulish, 1993).

Experience with goal-oriented measurement using the Balanced Scorecard (BSC) is presented in (Kaplan and Norton, 1996, 2000). In the next section, we look at a specific form of process measurement that has become increasingly popular: software process assessment.

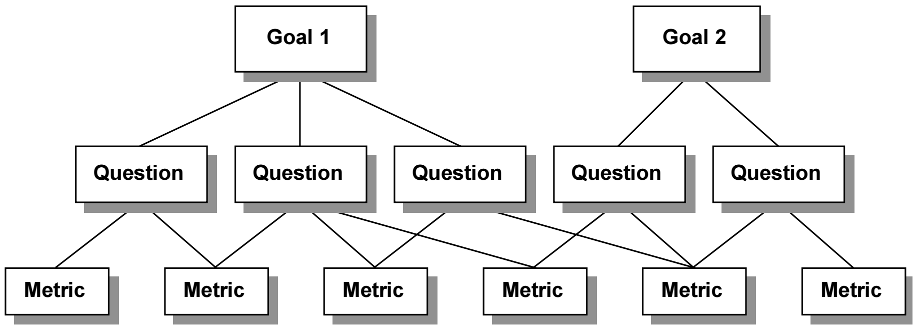


Figure 2.4: GQM hierarchical structure (Basili *et al*., 1994b).

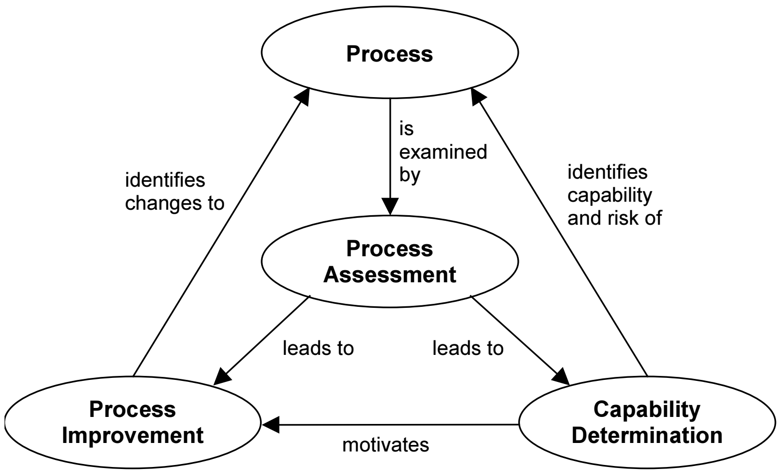


Figure 2.5: The context of software process assessment (ISO/IEC TR 15504-1:1998).

2.2 *Software Process Assessment*

A software process assessment (henceforth SPA) is an appraisal or review of a software organization to advice its management and professionals on how they can improve their operation (Humphrey, 1989). It can be used for process improvement or capability determination, as shown in *Figure 2.5*. General introductory overviews of process assessment are provided in (Humphrey, 1989; Zahran, 1998), an extensive empirical review is provided in (El Emam and Goldenson, 2000), and an overview of the application to SPI is provided in (Dybå, 1999).

2.2.1 *Assessment principles*

Basically, there are three ways in which a software organization can make an assessment of its software processes:

- Benchmark against other organizations.
- Benchmark against "best practice" models.
- Assessment guided by the specific goals and needs of the organization.

The first way of doing an assessment is a traditional benchmark exercise used to gain an outside perspective on practices and to borrow or "steal" ideas from best-in-class companies. This type of benchmarking can be seen as an ongoing investigation and learning experience that ensures that best practices are uncovered, analyzed, adopted, and implemented. An introductory overview of benchmarking is provided in (Camp, 1993). Also, the September/ October 2001 issue of *IEEE Software* had a focus on benchmarking of software organizations.

The second way of performing an assessment is to benchmark the company against one or more of the "best practice" models on the market (see next section for an overview). These models focus on different aspects of the software process and the organization, and they are all associated with specific strengths and weaknesses. Specifically, they have been criticized for being artificially derived and based on idealized lists of unvalidated practices (Fayad and Laitinen, 1997; Gray and Smith, 1998). They are associated with both statistical and methodological problems (Bollinger and McGowan, 1991), and they emphasize an improvement approach based on Statistical Process Control (SPC), which has been criticized by several authors (e.g. Dybå, 2000b; Ould, 1996).

During the 1990s, "software process assessment" has become synonymous with the model-based approach. The most well known method for performing such assessments is the CMM Based Appraisal for Internal Process Improvement (CBA IPI) (Dunaway and Masters, 1996). This method focuses on assessments for the purpose of process improvement using the Capability Maturity Model for Software (SW-CMM) (Paulk *et al.*, 1995). More recently, the Standard CMMI Assessment Method for Process Improvement (SCAMPI) (SEI, 2000a) has been developed for assessments relative to one or more of the CMM Integration (CMMI) models (e.g. SEI, 2000b, c). Furthermore, ISO/IEC CD 15504-2 (2001) provides guidance on how to perform a model-based assessment process according to the emerging standard (see *Figure 2.6*).

The third way of performing an assessment is with a participative approach tailored to the specific needs of the company, focusing on what is unique to each company and how this uniqueness can be exploited to gain competitive advantage. This approach is less time-consuming than the traditional benchmark approach, and it is clearly more relevant and valid than the model-based approach (Dybå and Moe, 1999).

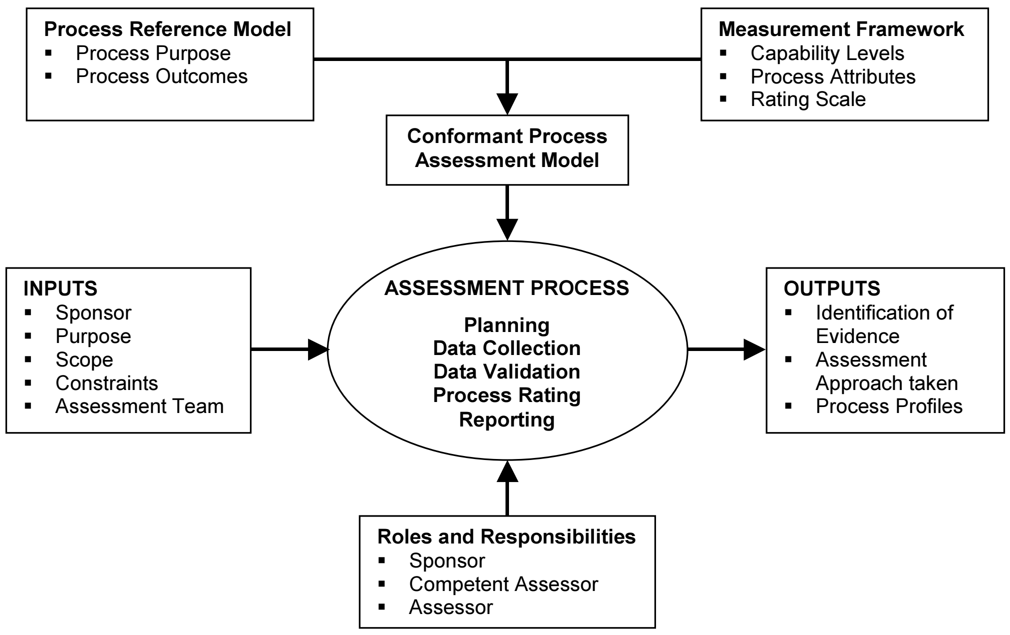


Figure 2.6: Model-based assessment process (ISO/IEC CD 15504-2: 2001).

Table 2.1: Model-based vs. participative approach to process assessment (Dybå and Moe, 1999).

Feature	**Model-based Approach**	**Participative Approach**
Focus areas and criteria from	"Best practices" according to the reference model.	Tailor made to the needs of the organization.
Data collected from	Selected group of managers and representatives from specific projects.	Everyone in the organization or department.
Data reported to	Sponsor (top management and department managers).	Everyone who participated (including management).
Role of researcher or consultant	Administration of questionnaires, documenting findings and recommendations.	Obtain agreement on assessment approach, joint design and administration of questionnaire, design of workshops.
Action planning done by	Top management.	Teams at all levels.
Probable extent of change and SPI	Low.	High.

The participative approach to software process assessment is part of the methodological basis used in SPIQ (Dybå, 2000e). The objective of this approach is to focus on the necessity of participation and tailoring in process assessment for SPI to take place. Basically, there are three reasons for this: (1) developers and managers alike must accept the data from the assessment as valid, (2) they must accept responsibility for the problems identified, and (3) they must start solving their problems (see *Section 8.2* for an overview of the approach and lessons learned from its use).

A summary and comparison of the model-based approach and the participative approach to software process assessment is given in *Table 2.1*.

2.2.2 *Assessment models*

The most commonly used assessment models in the software community is the SW-CMM (Paulk *et al.*, 1995) and the emerging international standard ISO/IEC 15504 (ISO/IEC CD 15504:2001). ISO/IEC 15504 is also known as SPICE – the name of the project developing the standard (El Emam *et al.*, 1998). Both of these initiatives had as their objective to improve the selection criteria's for potential software contractors to the U.S. and U.K. military, respectively, in order to reduce the risks associated with software projects and improve the quality of the delivered software.

Other notable examples of assessment models include ISO 9001 (2000) and its application to software (ISO/IEC WD 9000-3: 2001), TickIT (BSI, 2001), the EFQM Excellence Model (EFQM, 1999), Bootstrap (Kuvaja *et al.*, 1994), and Trillium (Coallier *et al.*, 1994).

Tingey (1996) compares and contrasts the Malcolm Baldridge national quality award, ISO 9000, and the CMM for software.

There is also a number of special purpose maturity models available, such as for systems engineering (Bate, 1995), software acquisition (Cooper *et al*., 1999), people management (Curtis *et al*., 1995), requirements engineering (Sommerville and Sawyer, 1997), testing (Burnstein *et al*., 1999), maintenance (Drew, 1992), measurement (Budlong and Peterson, 1995), and reuse (Karlsson, 1995). The applicability of assessment models to small organizations is addressed in (Johnson and Brodman, 1999; Sanders, 1998).

The architecture of "best practice" models can be classified as either staged or continuous (Paulk and Konrad, 1994). A *staged architecture*, like that of the SW-CMM, is uni-dimensional since it defines a set of Key Process Areas (KPAs) at each maturity level, except level 1 (see *Table 2.2*). A maturity level is defined in terms of satisfaction of the goals of the KPAs within the current (and the underlying) level.

The *continuous architecture*, on the other hand, like that of the ISO/IEC 15504, is two-dimensional (see *Figure 2.7*). One dimension consists of the processes that are actually assessed, while the second dimension consists of the capability scale that is used to assess process capability. Thus, *capability levels* apply to an organization's process-improvement achievement for *each* process area, while *maturity level*s apply to an organization's *overall* process capability and organizational maturity.

Table 2.2: Key Process Areas in the SW-CMM (Paulk *et al*., 1999).

Level	Focus	Key Process Areas
5 Optimizing	Continual process improvement	Defect Prevention Technology Change Management Process Change Management
4 Managed	Reduce process variation	Quantitative Process Management Software Quality Management
3 Defined	Standard engineering processes	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews
2 Repeatable	Basic project management	Requirements Management Software Project Planning Software Project Tracking and Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
1 Initial	Competent people and individual effort	None

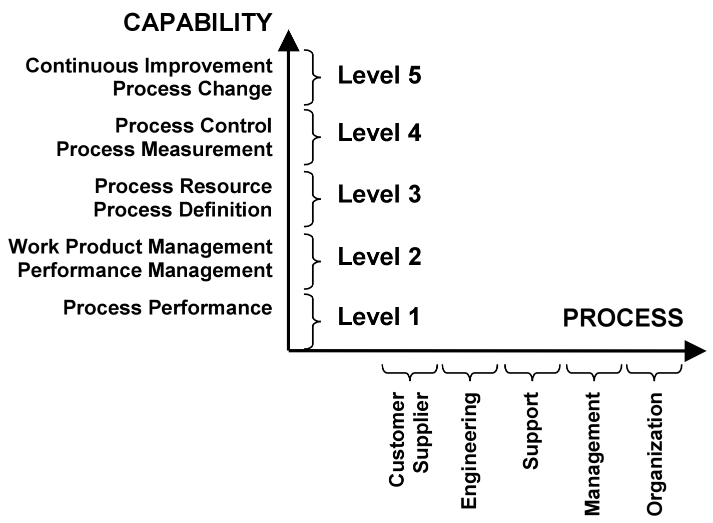


Figure 2.7: An overview of the ISO/IEC 15504 two-dimensional architecture (El Emam and Birk, 2000, p. 548).

Current work in the Software Engineering Institute's CMMI project aims to harmonize the two architectures, so that CMMI models can have both staged and continuous representations (e.g. SEI, 2000b, c).

2.2.3 *Assessment success factors*

While reliability and validity are necessary conditions for successful process assessment (El Emam and Birk, 2000; Fusaro *et al.*, 1998; Simon *et al.*, 1997), they are by no means sufficient (Dybå, 1999). Humphrey (1989), for example, identified a competent team, sound leadership, and a cooperative organization as the basic requirements for successful process assessments. Furthermore, he emphasized the need for a process model as a basis for the assessment, the requirement for confidentiality, senior management involvement, an attitude of respect for the views of the people in the organization being assessed, and, finally, an action orientation.

Also, based on a review of the SPA literature (e.g. Kuvaja *et al.*, 1994; Olson *et al.*, 1989; Paulk *et al.*, 1995; Pressman, 1988), Zahran (1998) mentioned the following critical success factors: Sponsorship and owner commitment, motivation, action orientation, confidentiality, relevance, credibility, and team building.

However, assessment alone does not create process change or improvement (see *Figure 2.1*). It just makes it possible and supports it. Therefore, to have effect beyond mere exploration, an assessment must be directed toward *action*. In this context, assessments can only be considered successful if they contribute to successful process improvement.

2.3 *Software Process Improvement*

A broad definition of SPI would include activities to (1) define and model a software process, (2) to assess the process, (3) to refine the process, and (4) to innovate a new process (see *Figure 2.1*). The fundamental assumption is that changes to the process cause improvements in the outcome. However, the term "process improvement" is ambiguous. Referring to the path diagram in *Figure 2.2*, it could mean either improvements in the process or improvements in the outcome.

General introductory overviews and guidelines for SPI are given in (e.g. Briand *et al.*, 1999; Cassidy and Guggenberger, 2000; Dybå, 2000e; Grady, 1997; Humphrey, 1989; Jeletic *et al.*, 1996; PROFES, 1999; Sanders, 1998; Zahran, 1998). Collections of theoretical and experiential SPI articles are provided in (El Emam *et al.*, 1998; El Emam and Madhavji, 1999; Hunter and Thayer, 2001; Messnarz and Tully, 1999).

Overviews on the evaluation of SPI and corresponding examples of empirical studies can be found in (El Emam and Briand, 1999; Goldenson *et al.*, 1999a; Jones, 1999; Kitchenham, 1996/1998; Krasner, 1999). A more pragmatic look at what can be achieved from such evaluation studies is provided by (Herbsleb, 1998).

2.3.1 *Improvement principles*

The Plan-Do-Check-Act (PDCA) cycle (*Figure 2.8*), developed by Walter Shewhart in the 1920s, provides the basic philosophy for a disciplined, cyclical approach to continuous improvement. PDCA is also referred to as the "scientific method" and the "Shewhart cycle". The cycle was later introduced by William Edwards Deming in his work with the Japanese industry after World War II.

While there are important differences, the ideas of quality or process improvement is just as applicable to software development as they are to manufacturing. However, quality management, as a discipline, is deeply rooted in the manufacturing process. The challenge, therefore, is to apply these general principles to a "development environment" instead of a "production environment" (Basili and Caldiera, 1995).

P – *Plan activities*, i.e. define the problem and state the improvement objectives.

D – *Implement the plan*, i.e. identify possible problem causes, establish baselines and test changes.

C – *Check the result*, i.e. collect and evaluate data.

A – *Improve the process*, i.e. implement system change and determine effectiveness.

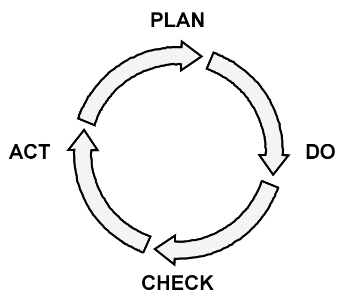


Figure 2.8: The Shewart improvement cycle (Deming, 1986).

Table 2.3: Benchmarking approach versus analytical approach (adapted from Jeletic *et al.*, 1996, p. 27).

Area	**Benchmarking Approach**	**Analytical Approach**
Goals	Focus on improving process	Focus on improving product
	Generalized goal (get to level 5)	Goals vary across organizations
	Domain independent	Domain dependent
	Common measure of success (higher level)	Measures of success vary from organization to organization
Initial Baseline	Perform assessment of process	Understand process and product
	Common yardstick (what is the maturity level?)	
Initial Analysis	Change process to advance to a higher level, reassess process (what is the maturity level now?)	Change process to improve product, reassess process and product
	Can compare across organizations?	Organization specific, no way to compare across organizations
Improvement Approach	Process based	Product based
	Common yardstick drives change	Organizational experience and goals drive change

Two basic approaches to SPI have emerged. Card (1991) referred to these approaches as the “benchmarking” and the “analytic” approach, while Thomas and McGarry (1994) used the terms “top-down” and “bottom-up” process improvement, respectively (see *Table 2.3* for a comparison).

The *benchmarking* approach compares an organization’s process with one of the “best practice” assessment models (see *Section 2.2.1*). Process improvement is then the elimination of differences between the existing process and the standard process. The assumption is that, once the process is changed the generated products will be improved – or at least the quality risks of generating new software will be reduced. The most well known benchmarking approach to SPI is the IDEAL model (McFeeley, 1996), which is based on transitioning the CMM into an organization’s practice (Peterson, 1995).

In contrast to the benchmarking approach, the *analytic* approach assumes that the organization’s individual goals, characteristics, product attributes, and experiences must drive process change; that change is defined by a local domain instead of a universal set of “best practices”. For example, an organization whose primary goal is improving time to market may take a significantly different approach to process change than one whose primary goal is to produce defect-free software. The most prominent example of the analytic approach to SPI is the Quality Improvement Paradigm (QIP) (Basili, 1989; Basili and Caldiera, 1995).

2.3.2 *Improvement models*

The most commonly used SPI models in the software community are the already mentioned IDEAL model and QIP. Other examples include the **ami** method (Pulford *et al.*, 1996), the project planning and development process (PADRE) (Rettig and Simons, 1993), and Grady's (1997) spiral model for process improvement adoption, which combines the Shewhart cycle with Boehm's (1988) spiral model of software development and enhancement. Also the SPIQ project (Dybå, 2000e) developed an improvement model based on the Shewhart cycle, while the PROFES project (PROFES, 1999) developed a model based on QIP.

Furthermore, the new part 4 of the ISO/IEC 15504 document set (ISO/IEC WD 15504-4:2001) provide guidance on utilizing process assessment for the purpose of process improvement and capability determination. It replaces the old ISO/IEC TR 15504-7:1998 and ISO/IEC TR 15504-8:1998. The guidance provides an eight-step process for continuous improvement, but does not presume specific organizational structures, management philosophies, software life cycle models, or software development methods.

Of the improvement models, IDEAL most clearly locates specific learning activities in the last phase – the learning phase. However, it is our contention that learning should not reside in one phase, as an after-the-fact reflection. Rather, learning should be recognized as an integral part of all phases of the improvement cycle (see *Chapters 4*, *5*, and *9*).

2.3.2.1 The IDEAL model

The SEI's recommended framework for software process improvement is the IDEAL model (McFeeley, 1996) shown in *Figure 2.9*. The IDEAL model was developed in order to present a consistent view of the activities of an improvement program based on transitioning the CMM into an organization's practice (Peterson, 1995).

The IDEAL approach consists of the following five phases (Gremba and Myers, 1997):

(1) **Initiating** (the improvement program). The Initiating phase establishes the business reasons for undertaking a software process improvement effort. It identifies high-level concerns and business goals that can be the stimulus for addressing various aspects of quality improvement. Communication of these concerns and business perspectives is needed during the Initiating phase in order to gain executive commitment and sponsorship at this very early part of the improvement effort.

(2) **Diagnosing** (the current state of practice). The Diagnosing phase builds on the initiating phase to develop a common understanding of the strengths and weaknesses of the current processes of the organization, and to help identify priorities for improving the software processes. This diagnosis is based on the CMM model and the SCAMPI method.

(3) **Establishing** (the plans for the improvement program). The Establishing phase finalizes the strategy and supporting plans for the software process improvement program. It sets the direction and guidance for short-term and long-term improvement, including strategic and tactical plans for SPI.

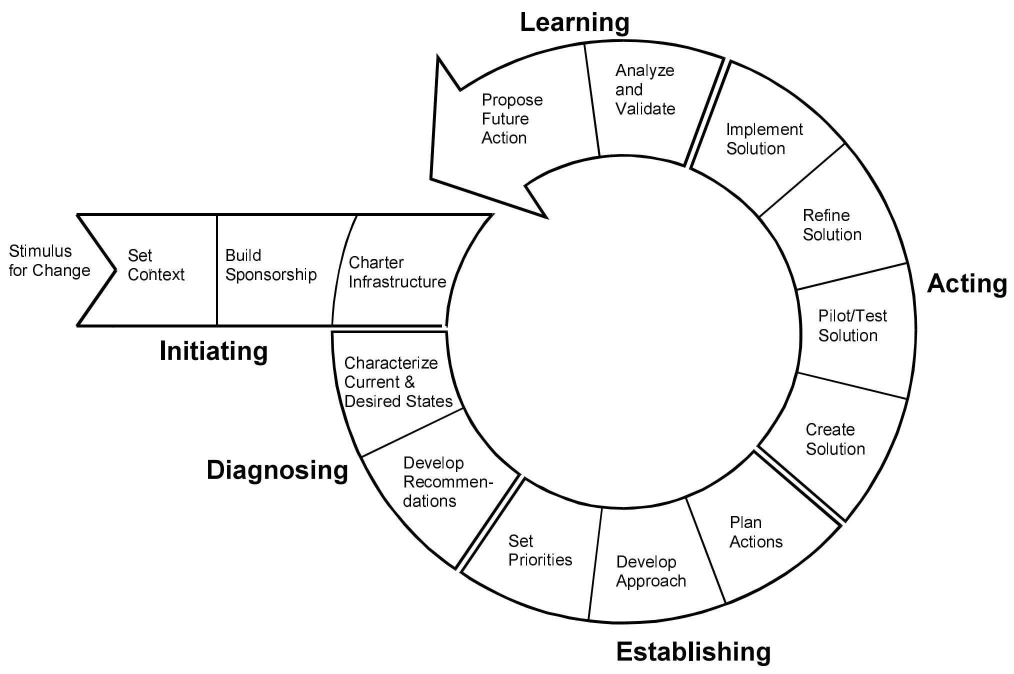


Figure 2.9: The IDEAL v1.1 model (Gremba and Myers, 1997).

(4) **Acting** (on the plans and recommended improvements). The Acting phase takes action to effect changes in organizational systems that result in improvements in these systems. These improvements are made in an orderly manner and in ways that will cause them to be sustained over time. Techniques used to support and institutionalize change include defining software processes and measurements, pilot testing, and installing new processes and measurements throughout the organization.

(5) **Learning** (from the lessons and the business results of the improvement effort). The Learning phase completes the process improvement cycle. Lessons learned from the pilot projects and improvement efforts are documented and analyzed in order to improve the process improvement program for the future. The business needs that were determined at the beginning of the cycle are revisited to see if they have been met. Sponsorship for the program is revisited and renewed for the next cycle of software process improvement.

2.3.2.2 The Quality Improvement Paradigm

The Quality Improvement Paradigm (QIP) developed by Basili *et al.* (Basili, 1989; Basili and Rombach, 1991; Basili and Caldiera, 1995; Basili *et al.*, 1994a), is the result of the application of the Shewhart cycle to the problem of software quality improvement.

QIP consists of the following six steps:

(1) **Characterize**. Understand the environment based upon available models, data, intuition, etc. Establish baselines with the existing business processes in the organization and characterize their criticality.

(2) **Set goals**. On the basis of the initial characterization and of the capabilities that have a strategic relevance to the organization, set quantifiable goals for successful project and organization performance and improvement. The reasonable expectations are defined based upon the baseline provided by the characterization.

(3) **Choose process**. On the basis of the characterization of the environment and of the goals that have been set, choose the appropriate processes for improvement, and supporting methods and tools, making sure that they are consistent with the goals that have been set.

(4) **Execute**. Perform the processes constructing the products and providing project feedback based upon the data on goal achievement that are being collected.

(5) **Analyze**. At the end of each specific project, analyze the data and the information gathered to evaluate the current practices, determine problems, record findings, and make recommendations for future project improvements.

(6) **Package**. Consolidate the experience gained in the form of new, or updated and refined, models and other forms of structured knowledge gained from this and prior projects, and store it in an experience base so it is available for future projects.

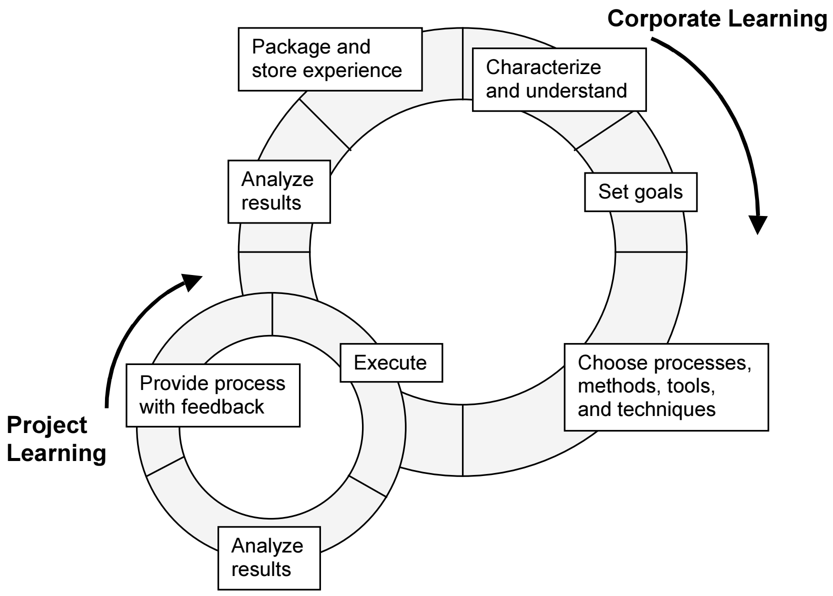

Corporate Learning
Package and store experience
Characterize and understand
Analyze results
Set goals
Execute
Provide process with feedback
Choose processes, methods, tools, and techniques
Project Learning
Analyze results

Figure 2.10: The Quality Improvement Paradigm (Basili and Caldiera, 1995, p. 58).

QIP implements two feedback cycles (see *Figure 2.10*), which were also used in SPIQ (Dybå, 2000e):

- The **project feedback (or control) cycle** is the feedback that is provided to the project during the execution phase. It provides analytic information about project performance at intermediate stages of development by comparing project data with the nominal range for similar projects.
- The **corporate feedback (or capitalization) cycle** is the feedback that is provided to the organization. Its purpose is to understand what happened, by capturing experience across application domains and to accumulate reusable experience in the form of experience packages.

The GQM approach, as discussed in the previous section, is the mechanism used by QIP for defining and evaluating a set of operational goals using measurement. In addition to the GQM approach, QIP uses the Experience Factory organization for building software competencies and supplying them to projects.

2.3.3 *Improvement success factors*

The current state-of-the-art in quality management has more than anything else been shaped by quality gurus such as William Edwards Deming (1982, 1986), Joseph M. Juran (1992; Juran and Godfrey, 1999), Philip Crosby (1979, 1984, 1996), and their quality frameworks. *Table 2.4* summarizes Deming's 14 principles, the Juran trilogy, and Crosby's 14 quality steps. These and other authors (e.g. Ahire *et al.*, 1996; Black and Porter, 1996; Feigenbaum, 1991; Garvin 1983, 1984; Ishikawa, 1986, 1990; Powell, 1995; Saraph *et al.*, 1989; Taguchi, 1986; Taguchi *et al.*, 1989; Yusof and Aspinwall, 1999) repeatedly discuss the importance of critical factors such as leadership involvement, employee participation, measurement, and process management to improve the quality performance in organizations. For a more detailed review of these, and other empirical studies of the factors for success in quality management, organizational learning, and SPI, see (Dybå, 2000a).

Within SPI, Humphrey (1989) identified six basic principles of software process change. Zahran (1998) proposed ten critical factors for successful implementation of software process improvement. Basili and Caldiera (1995) focused on reuse of experience and learning by using QIP for developing core competencies, and by supporting the QIP process with goal-oriented measurement using GQM and with an organizational infrastructure based on EF. *Table 2.5* summarizes these concepts.

Goldenson and Herbsleb (1995) conducted a survey of 138 individuals from 56 organizations in the United States and Canada to evaluate organizational factors that were believed to promote or hinder successful SPI after a CMM-based assessment. The factors that were found to be statistically significant in their study are summarized in *Table 2.6*.

El Emam *et al*. (2001) made a reanalysis of Goldenson and Herbsleb's (1995) study, using multivariate analysis instead of the simple statistical analytic methods used in the initial report. Based on this reanalysis, they identified focused SPI effort, commitment to SPI, politics, respect, and turnover as the key factors.

Table 2.4: Perspectives on quality management.

Deming's 14 principles[1]	**The Juran Trilogy**[2]	**Crosby's 14 quality steps**[3]
1. Constancy of purpose	I. *Quality Planning*	1. Management commitment
2. Adopt the new philosophy	Establish quality goals	2. Quality improvement teams
3. Cease dependence on inspection	Identify customers and their needs	3. Quality measurement
4. Don't award business on price	Develop products and processes	4. Cost of quality evaluation
5. Constant improvement	II. *Quality Control*	5. Quality awareness
6. Institute training on the job	Evaluate performance	6. Corrective action
7. Institute leadership	Compare to goals and act	7. Zero-defects committee
8. Drive out fear	III. *Quality Improvement*	8. Supervisor training
9. Break down barriers	Establish infrastructure	9. Zero-defects day
10. Eliminate slogans	Identify improvement projects and teams	10. Goal-setting
11. Eliminate work standards	Provide resources and training	11. Error cause removal
12. Pride of workmanship	Establish controls	12. Recognition
13. Education and retraining		13. Quality councils
14. Take action		14. Do it over again

Sources: [1]Deming (1986), [2]Juran (1992), [3]Crosby (1979).

Table 2.5: Facilitating factors identified in software process improvement.

Humphrey's six principles[1]	**Zahran's 10 CSFs**[2]	**Basili's paradigm**[3]
• Major changes to the software process must start at the top	1. Alignment with the business strategy and goals	• Acquisition of core competencies through (1) a control cycle and (2) a capitalization cycle
• Ultimately, everyone must be involved	2. Consensus and buy-in from all stakeholders	• Goal-oriented measurement
• Effective change is built on knowledge	3. Management support	• Experience reuse and organizational sharing
• Change is continuous	4. Dedicated resources	
• Software process changes won't stick by themselves	5. Sensitivity to the organizational context	
• Software process improvement requires investment	6. Management of change	
	7. Prioritization of actions	
	8. Support infrastructure	
	9. Monitoring the results of SPI	
	10. Learning from the feedback results	

Sources: [1]Humphrey (1989), [2]Zahran (1998), [3]Basili and Caldiera (1995).

Table 2.6: Facilitating factors and barriers to SPI (Goldenson and Herbsleb, 1995).

Organizational Factors	Barriers
• Senior management monitoring of SPI	• Discouragement about SPI prospects
• Compensated SPI responsibilities	• SPI gets in the way of “real” work
• SPI goals well understood	• “Turf guarding” inhibits SPI
• Technical staff involved in SPI	• Existence of organizational politics
• SPI people well respected	• Assessment recommendations too ambitious
• Staff time/resources dedicated to process improvement	• Need guidance about how to improve
	• Need more mentoring and assistance

Within the SPICE Trials, a similar study to that of Goldenson and Herbsleb (1995) was conducted by El Emam *et al*. (1999) with 18 organizations in Europe, Canada, and Australia that had performed assessments using the ISO/IEC 15504 standard for software process assessment. In their study, three types of independent variables were tested: “organizational factors,” “process factors,” and “barriers”.

Results of the bivariate relationship analysis showed that none of the identified barriers were related to success in addressing the findings from an assessment. Of the organizational factors, only “SPI goals being well understood” and “Technical Staff involvement in SPI” were found to be critical for addressing the findings from an assessment. Finally, only one process factor, “Creating process action teams”, was found to be statistically significant in addressing the assessment findings.

Stelzer *et al*. (1996) identified the following key success factors in their study of software process improvement via ISO 9000: (1) definition and documentation of the status quo, (2) identification of best practices, (3) identification of business processes, (4) simplification of routine procedures, (5) internal audits, (6) impetus and incentive, (7) team spirit, (8) workshop and regular meetings, (9) definition of a common language, and (10) customer perception surveys.

Furthermore, Stelzer and Mellis (1998) analyzed published experience reports and case studies of 56 software organizations that had implemented an ISO 9000 quality system or that had conducted a CMM-based SPI initiative. The result of this meta-analysis was a set of ten factors that affect organizational change in SPI. In rank order, these factors were: (1) management commitment and support, (2) staff involvement, (3) providing enhanced understanding, (4) tailoring improvement initiatives, (5) managing the improvement project, (6) change agents and opinion leaders, (7) stabilizing changed processes, (8) encouraging communication and collaboration, (9) setting relevant and realistic objectives, and (10) unfreezing the organization.

Moreover, in a survey of 87 projects from different organizations, Deephouse *et al*. (1996) assessed the effectiveness of software processes on project performance. The results from this study showed that certain practices, such as project planning and cross-functional teams, were consistently associated with favorable outcomes, while other practices such as process training, stable environment, user contact, design reviews, and prototyping had little impact on project outcomes.

Finally, ISO/IEC WD 15504-4: 2001 highlight cultural issues as fundamental to succeed with software process improvement and organizational change. The standard argues that SPI should be strongly supported by leadership, communication, and motivation throughout the whole organization and that the major problems found in software processes often arise from cultural issues. Consequently, cultural issues should be one of the factors considered in prioritizing improvement actions.

2.4 Software Process Innovation

Process improvement strategies, as implemented by the SPI approaches described in the previous section, represent relatively small, evolutionary, or incremental changes, in the organization's products, procedures, or services. They are new to the organization but reflect an adaptation or simple adjustment of existing practices, and their implementation rarely requires changes in organizational structures or processes. In contrast, *process innovation* strategies represent larger changes in organizational products, procedures, or services. They reflect broader shifts in perspective and reorientation of existing practices and often require major or radical changes in organizational structures or processes to implement.

The distinction between incremental and radical change is also consistent with a characterization of learning strategies ranging from adaptive to innovative. Organizations with adaptive styles work within existing structures to make incremental changes and "do things better". In contrast, organizations with innovative styles treat current structures as part of the problem and make more radical changes by "doing things differently".

A review of innovation research can be found in (Gopalakrishnan and Damanpour, 1997). An integrated set of studies of the innovation process in organizations, which resulted from the Minnesota Innovation Research Program, is provided by (Poole *et al.*, 2000; van de Ven *et al.*, 1999; van de Ven *et al.*, 2000). More management oriented overviews and guidelines on the innovation process can be found in (Leifer *et al.*, 2001; Leonard-Barton, 1995; Nonaka and Takeuchi, 1995). Furthermore, technological innovation as an evolutionary process is provided by (Ziman, 2000).

General guidance on the diffusion of innovations can be found in Rogers' (1995) classic text on the subject. Software engineering books and articles that discuss the diffusion of innovations include (Ardis and Marcolin, 2001; Pfleeger, 1999b; Pfleeger and Menezes, 2000; Raghavan and Chand, 1989). Also, Kautz and Larsen (2000) described a European-wide project for the dissemination of quality management and SPI innovations.

Furthermore, various theoretical perspectives have described the nature of the relationship between adaptation and innovation from an organizational learning perspective (e.g. Argyris and Schön, 1996; Fiol and Lyles, 1985; Lant and Mezias, 1992; March, 1999; Senge, 1990; Tushman and Romanelli, 1985).

2.4.1 Innovation principles

The common thread that runs through the literature on innovation is "newness"; on the most basic level, innovation means "something new" (Gopalakrishnan and Damanpour, 1997). Building on the organizational learning literature (e.g. Argyris and Schön, 1996), process innovation can be defined as higher-level learning that results in new work practices as well as a restructuring of norms, strategies, and assumptions governing these practices.

Such process innovations can also be seen in business process reengineering (BPR) and redesign projects (Davenport, 1993; Hammer, 1990). Hammer and Champy (1993), for example, described the principle behind "reengineering" as "the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service, and speed." (*ibid*., p. 32).

Davenport (1993), preferring the term "process innovation", stated that "Reengineering is only part of what is necessary in the radical change of processes; it refers explicitly to the design of the new process. The term process innovation encompasses the envisioning of new work strategies, the actual process design activity, and the implementation of the change in all its complex technological, human, and organizational dimensions." (*ibid*., p. 2).

Based on our review of the innovation, reengineering, and organizational learning literature, the following principles can be seen as fundamental to process innovation:

(1) **Process**. The distinctive element is business process or work practice. Referring to the software engineering domain, processes can be categorized as "primary", "supporting", or "organizational" (ISO/IEC 12207: 1995).

(2) **Transformation**. The typical promise of process innovation is that a quantum leap in organizational performance is available and that the organization will have to identify or create suitable innovations to achieve this goal (e.g. Earl, 1996; O'Neill and Sohal, 1999).

(3) **Enabling conditions**. The classic texts on BPR have focused on information technology (IT) as the most important enabler for radical change. However, autonomy, diversity, flexibility, and the ability to question established "truths" have also been noted as primary drivers in creating the dynamics of innovation (e.g. Argyris and Schön, 1996; Nonaka and Takeuchi, 1995)

(4) **Change management**. A key challenge in process innovation is change management, which includes the people, technology, norms, assumptions, and strategy, along with planning, structuring, and evaluation of process innovations (e.g. Grover, 1999).

Furthermore, van de Ven (1986) focused on the social dynamics of innovation, noting that:

> Innovation is not the enterprise of a single entrepreneur. Instead, it is a network-building effort that centers on the creation, adoption, and sustained implementation of a set of ideas among people who, through transactions, become sufficiently committed to these ideas to transform them into 'good currency' (*ibid*., p. 601).

2.4.2 Innovation models

Innovation models typically rely on a series of stages from broader, general ideas that become more formalized and specialized to form a specific innovation. Marquis (1988), for example, modeled the successful process of innovation as a series of six steps ranging from recognition to utilization and diffusion. Similarly, Rogers (1995) defined the innovation-development process as the decisions and activities that occur from recognition of a need or a problem, through research, development, and commercialization, through diffusion and adoption of the innovation by users, to its consequences.

Another body of literature expresses the innovation process less as a set of sequential stages, but more as an iterative "nonlinear" approach. Lynn *et al.* (1996), for example, described a "probe-and-learn" process for opportunity analysis of innovations. Others have stated that innovations are born of chaotic, nonlinear processes with no set stages (e.g., Jelinek and Schoonhoven, 1990; Kline, 1985; Quinn, 1985).

Furthermore, van de Ven *et al.* (1999) presented the results of the Minnesota Innovation Research Program, a major longitudinal study that examined the process of innovation from concept to implementation of new technologies, products, and processes. According to their findings, the innovation journey is neither sequential and orderly, nor is it a matter of random trial and error. Rather it is best characterized as a nonlinear dynamic system. This system consists of a cycle of divergent and convergent activities, which was found to be the underlying dynamic that explained the development of corporate cultures for innovation.

Similarly, based on a case study on the introduction of innovative organizational processes, Presley *et al.* (2000) recommended an approach for product and process innovation based on the Soft-Systems Methodology (Checkland, 1981, 1999).

In the following subsections, we present Davenport's (1993) general approach to process innovation and Pfleeger's (1999b) approach to the diffusion of innovations in software engineering.

2.4.2.1 Davenport's approach to process innovation

Davenport's (1993) approach to process innovation combines the adoption of a process view of the business with the application of innovation to key processes in order to create radical change. The process innovation approach consists of five major steps (see *Figure 2.11*):

(1) **Identifying processes for innovation**. Process innovation begins with identifying the processes that are candidates for innovation, assessing their strategic relevance, and establishing the boundaries of the processes that are to be addressed.

(2) **Identifying change levers**. Change lever analysis relies on both knowledge and creative thinking about how IT and innovative organizational/human resource approaches might be applied to the process under analysis. This requires knowledge about the latest state and likely future capabilities of key technologies or human enablers of change, and knowledge about the ways these change enablers have been or could be applied to the target process.

(3) **Developing process visions**. Developing a worthwhile process vision relies on a clear understanding of organizational strengths and weaknesses, coupled with an understanding of market structure and opportunity. Also, it requires knowledge about innovative activities undertaken by competitors and other organizations.

(4) **Understanding existing processes**. Understanding existing processes are important before designing a new one. It facilitates communication among participants in the innovation initiative, it helps ensure that existing problems are not repeated in the new process, and it provides a measure of the value of the proposed innovation.

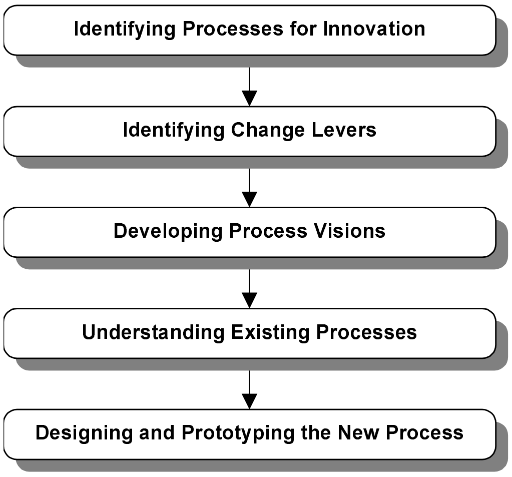

Identifying Processes for Innovation
Identifying Change Levers
Developing Process Visions
Understanding Existing Processes
Designing and Prototyping the New Process

Figure 2.11: Davenport's approach to process innovation (Davenport, 1993, p. 25).

(5) **Designing and prototyping the new process**. The design process begins by building on the high-level process concept developed during the visioning stage, shifting toward a detailed process design. Prototyping enables design possibilities to be presented, approved, and/or modified by users before the organization has invested much effort in detailed design and implementation.

2.4.2.2 Pfleeger's approach to technology transfer in software engineering

Most innovation models, like Davenport's (1993) approach outlined above, is only the first step toward full-scale implementation of new processes. The diffusion and implementation of innovation, which is also called "technology transfer", is best seen in the work of Rogers (1995) who viewed innovations in the broader context of organizational change, focusing on how they can be successfully adopted.

Based on Rogers' work on the diffusion of innovations, Pfleeger (1999b) defined a model of technology transfer in software engineering. In this model, "technology" means any method, technique, tool, procedure, or paradigm used in software development or maintenance.

The technology transfer process consists of five steps (see *Figure 2.12*):

(1) **Technology creation**. Technology creation begins with a business need and asks whether a technology exists that might address it. Such technology might exist and be in use somewhere else, it might exist but is untried on this problem, or it might not exist and must thus be created.

(2) **Technology evaluation: preliminary**. Once a candidate technology is found, the next step is a preliminary investigation to determine whether there is evidence that the technology will work in practice.

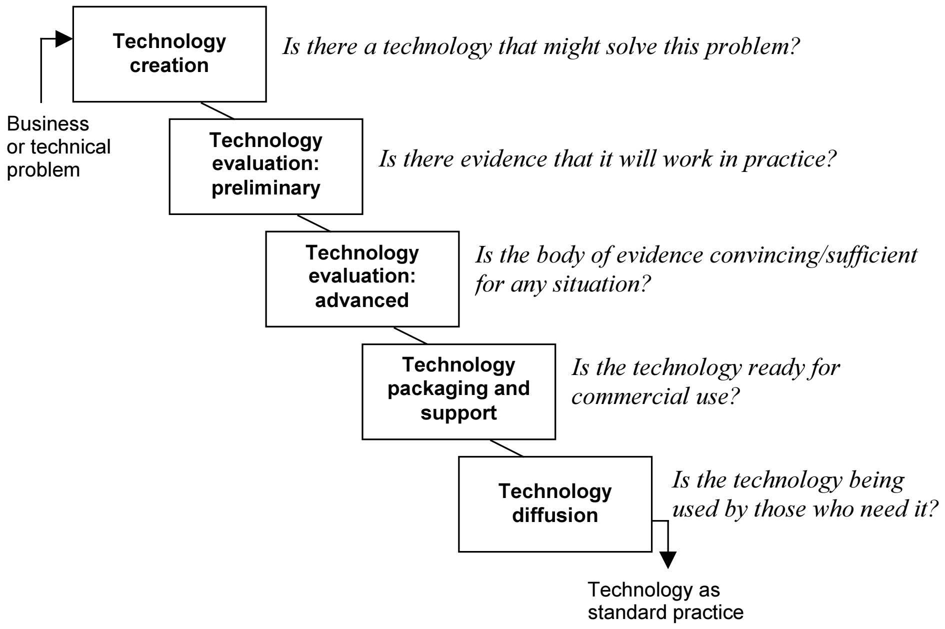

Technology creation
Is there a technology that might solve this problem?
Business or technical problem
Technology evaluation: preliminary
Is there evidence that it will work in practice?
Technology evaluation: advanced
Is the body of evidence convincing/sufficient for any situation?
Technology packaging and support
Is the technology ready for commercial use?
Technology diffusion
Is the technology being used by those who need it?
Technology as standard practice

Figure 2.12: Pfleeger's technology transfer process (Pfleeger, 1999b, p.116).

(3) **Technology evaluation: advanced**. Once the evidence indicates that the technology has worked in practice, the next step is a more thorough evaluation of the body of evidence. Issues of interest to this step include the situations in which the technology worked and the methodological nature of the studies.

(4) **Technology packaging and support**. If the technology continues to look promising, the next step is to look at tools and other packaging and support to aid the technology's use.

(5) **Technology diffuison**. Once there is convincing evidence of the effectiveness of the new technology, plus appropriate packaging and support, the technology can be transferred to a wider audience.

2.4.3 *Innovation success factors*

Rogers (1995) has shown that the rate of adoption of an innovation is determined by the characteristics of an innovation *as perceived by the potential adopter*, and not whether it has produced any advantages for competitors. His research has shown that the diffusion of innovations depends on the following five factors:

(1) **Relative advantage** – the degree to which an innovation is perceived as being better than the idea it supersedes.

(2) **Compatibility** – the degree to which an innovation is perceived as consistent with the existing values, past experience, and needs of potential adopters.

(3) **Complexity** – the degree to which an innovation is perceived as relatively difficult to understand and use.

(4) **Trialability** – the degree to which an innovation may be experimented with on a limited basis.

(5) **Observability** – the degree to which the results of an innovation are visible to others.

Furthermore, diffusion of innovation models emphasize the importance of *homophily*, which Rogers (1995) defined as the degree to which the innovator and the potential adopter are similar in certain attributes such as objectives, beliefs, norms, experience, and culture. Heterophily is the opposite of homophily, and, according to Rogers (1995), "*one of the most distinctive problems in the diffusion of innovations is that the participants are usually quite heterophilous*." (*ibid*., p. 19, italics in original). Hence, vital differences between innovators and potential adopters act as key barriers to imitation (Powell, 1995).

Finally, there is an increasing recognition that successful innovation and technology transfer requires an orientation toward organizational learning and knowledge creation (e.g. Levin, 1997; Nonaka and Takeuchi, 1995). This is also, as we shall see, the position taken in this thesis regarding process improvement and process innovation in software organizations.

2.4.4 *Process improvement versus process innovation*

There are important similarities as well as differences between process improvement and process innovation. These similarities and differences can be illustrated by comparing TQM with BPR. During the 1990s we have witnessed an intense debate between the two schools of organizational change. *Table 2.7* shows the main differences between TQM (process improvement) and BPR (process innovation), as pointed out by Davenport (1993).

Table 2.7: Process improvement versus process innovation (Davenport, 1993, p. 11).

	Process Improvement	**Process Innovation**
Level of Change	Incremental	Radical
Starting Point	Existing process	Clean slate
Frequency of Change	One-time/continuous	One-time
Time Required	Short	Long
Participation	Bottom-up	Top-down
Typical scope	Narrow, within functions	Broad, cross-functional
Risk	Moderate	High
Primary Enabler	Statistical control	Information technology
Type of Change	Cultural	Cultural/structural

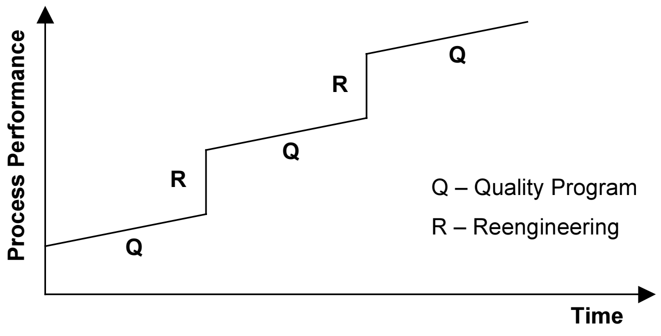


Figure 2.13: Process improvement and process innovation (Hammer, 1996, p. 83).

Although, process improvement and process innovation strategies are clearly different, they, nevertheless, share important similarities – a fact that the founders of BPR have always stressed (Davenport, 1993; Hammer and Champy, 1993; Hammer, 1996). Also, process innovation was encouraged by the early quality experts (e.g. Juran, 1964). Both approaches focus on processes, both start with the customer's needs and work backwards from there, and both recognize the importance of teamwork and cultural change. In a similar vein, Brown and Duguid (1991) emphasized a unified view of working, learning, and innovation.

Figure 2.13 illustrates how process improvement and process innovation fit together over time in the life-cycle of a process. First, the process is enhanced until its useful lifetime is over, at which point it is reengineered. Then, enhancement is resumed and the entire cycle starts again (Hammer, 1996). It is important to note, here, that improvement and innovation are complementary approaches to SPI, and that both forms are necessary for software organizations (Dybå, 2000b). This situation, which requires the management of both stability and change, has also been described with a punctuated equilibrium model (Tushman and Romanelli, 1985; Lant and Mezias, 1992).

From this perspective, we propose a high-level approach for SPI that encompasses process definition, process assessment, process improvement, and process innovation (see *Figure 2.14*). The major features of this approach are an appropriate understanding, definition, and selection of processes, either for improvement or innovation, according to the organization's vision and goals, with feedback based on the measurement or assessment of results.

2.5 Chapter Summary

We started this chapter by describing the concepts of software process. Based on these concepts, we then described three intervention strategies or approaches to change the software process in order to bring about improvements in organizational performance. First, *process assessment* was described as an intervention strategy that involves an appraisal or review of an organization's software processes without changing them. Second, we described *process improvement* as an intervention strategy that involves continuous or incremental change to the organization and its processes. Finally, we described *process innovation* as an intervention strategy that involves radical change to the organization with replacement of its processes.

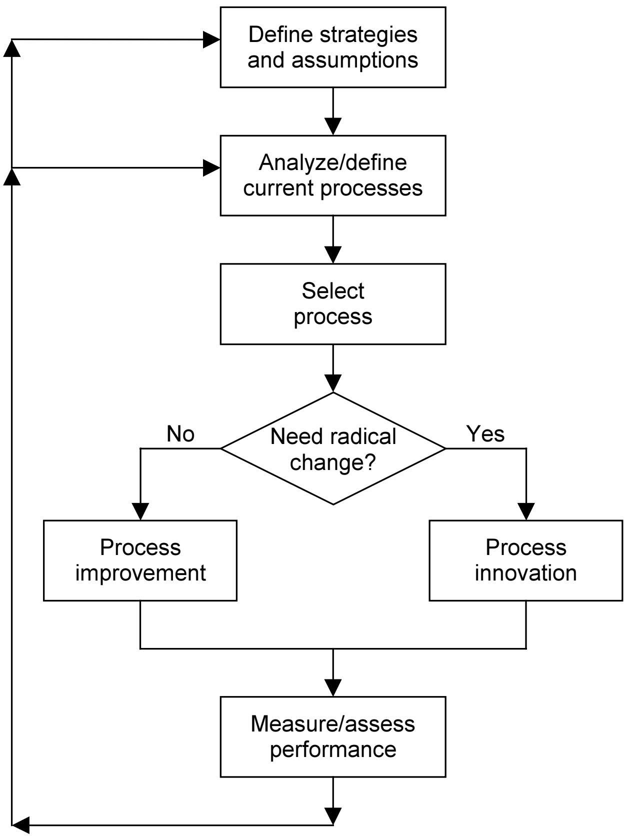

Define strategies and assumptions
Analyze/define current processes
Select process
Need radical change?
No
Yes
Process improvement
Process innovation
Measure/assess performance

Figure 2.14: A high-level proposal for successful SPI that integrates process definition, process assessment, process improvement, and process innovation.

In the next chapter, we take a closer look at empirical software engineering research and the major research approaches that are applicable for investigations in SPI. Four classes of such approaches are reviewed: experimental research, survey research, case study research, and action research. In addition, the chapter presents a comparison of the key features, strengths, and weaknesses of each of these research approaches.

CHAPTER 3

Approaches to Empirical Software Engineering Research

"*I believe that the present fashion of applying the methods of physics to human life is not only a mistake but heinous.*"

– Albert Einstein

In the previous chapter, we described the concepts of software process and three intervention strategies or approaches to SPI. First, *process assessment* was described as an intervention strategy that involves an appraisal or review of an organization's software processes without changing them. Second, we described *process improvement* as an intervention strategy that involves continuous or incremental change to the organization and its processes. Third, we described *process innovation* as an intervention strategy that involves radical change to the organization with replacement of its processes. Finally, we made a high-level proposal for successful SPI that integrates process definition, process assessment, process improvement, and process innovation.

From our point of view, there is a close relationship between approaches to SPI, specifically those that are based on the scientific method, and approaches to empirical software engineering (henceforth ESE) research. Both are concerned with the investigation of software practice and both are concerned with improving it. The main difference is that the SPI approaches primarily are used by practitioners, while the ESE research approaches primarily are used by researchers.

In this chapter, therefore, we take a closer look at ESE research and the major approaches that are relevant for investigations in SPI. Four classes of such research approaches will be reviewed: experimental research, survey research, case study research, and action research. In addition, the chapter presents a comparison of the features, strengths, and weaknesses of each of these approaches.

3.1 Empirical Software Engineering Research

ESE research can be defined as analysis based on the investigation of actual practice for the purpose of discovering the unknown or testing a hypothesis. It involves an investigator gathering data and performing analysis to determine the meaning of the data. Rather than empirical research, however, computer scientists generally tend to adhere to a more "ad-hoc" evaluation of their research, advocating and publishing ideas with little or no scientific assessment (Basili, 1996; Fenton *et al.*, 1994; Fuggetta, 1999; Glass, 1994; Potts, 1993; Tichy, 1998).

Tichy *et al.* (1995), for example, made a survey of 400 computer science research papers published by the ACM to investigate their rigor. Considering those papers whose claims required empirical evaluation, they found that 40 percent of these papers had no empirical support at all. In software-related journals, the fraction was 50 percent. For comparison, in the other disciplines that were considered, the figure was merely 15 percent.

The study by Zelkowitz and Wallace (1998) found similar results. In their study of 562 software engineering research papers published by the IEEE, they found that the most prevalent validation models were lessons learned and case studies, each at a level of about 10 percent. About a third of the papers, however, had no empirical validation. Furthermore, Zelkowitz and Wallace (1998) observed that the authors often failed to state their goals clearly or to point out the value of their methods and tools to the empirical validation, that they often failed to state how they validated their hypotheses, and that they often used terms very loosely.

ESE research seeks to address this deficiency by encouraging a more scientific and, in our view, reflective approach to software engineering, which allows us to investigate and understand new technologies and their applications to practice.

A growing interest in ESE research is also reflected by the number of initiatives devoted specifically to the role of empirical studies as a means for improving software engineering research and practice, e.g.:

- *An international research network*, called "International Software Engineering Research Network (ISERN)", which was established in 1993 (see www.iese.fhg.de/ISERN).
- *An international journal*, entitled "Empirical Software Engineering: An International Journal", which was launched by Kluwer Academic Publishers in 1996 with Victor R. Basili and Warren Harrison as the editors-in-chief (see kapis.www.wkap.nl/kapis/CGI-BIN/WORLD/journalhome.htm?1382-3256).
- *An international conference*, called "Empirical Assessment in Software Engineering (EASE)", which was created at Keele University, U.K. and held for the first time in 1997 (see www.keele.ac.uk/depts/cs/ease).
- *A center for empirically based software engineering*, called CeBASE, which was organized in 2000 (see www.cebase.org).

3.1.1 Dimensions of empirical software engineering research

ESE research comes in several shapes and sizes. Before a study begins, the researcher must make several decisions, such as deciding the purpose of the study, the main approach to be taken, whether it should be based on qualitative or quantitative methods, and how the time dimension should be treated. By understanding these dimensions of ESE research, the software researcher will be better prepared to make decisions about the conduct of his or her investigation.

3.1.1.1 The purpose of the study

The main purpose of a study can be organized into three groups based on what the researcher is trying to accomplish. That is, either to *explore* a new topic, to *describe* a phenomenon, or to *explain* why something occurs (see *Table 3.1*).

Table 3.1: Purpose of research (Neuman, 2000, p. 22).

Exploratory	**Descriptive**	**Explanatory**
• Become familiar with the basic facts, setting, and concerns • Create a general mental picture of conditions • Formulate and focus questions for future research • Generate new ideas, conjectures, or hypotheses • Determine the feasibility of conducting research • Develop techniques for measuring and locating future data	• Provide a detailed, highly accurate picture • Locate new data that contradict past data • Create a set of categories or classify types • Clarify a sequence of steps or stages • Document a causal process or mechanism • Report on the background or context of a situation	• Test a theory's predictions or principle • Elaborate and enrich a theory's explanation • Extend a theory to new issues or topics • Support or refute an explanation or prediction • Link issues or topics with a general principle • Determine which of several explanations is best

3.1.1.2 The theory-building-and-testing dimension

The building and testing of theories in ESE research can be approached from two directions (see *Figure 3.1*). The *inductive* approach begins with detailed observations of the world, moving toward abstract generalizations and theories, and eventually to the formulation of hypotheses. An example of such an inductive approach to theory building is grounded theory (Strauss and Corbin, 1998).

The other, *deductive*, approach begins with an abstract, logical relationship among concepts, formulated as a hypothesis, and then moves toward concrete empirical evidence by testing the hypothesis. The evidence can either lead to a confirmation or falsification of the hypothesis and, subsequently, to refinements to the theory. This hypothetico-deductive approach is at the heart of all experimental research (e.g. Wohlin *et al.*, 2000). In practice, however, most ESE research projects use both approaches during the study.

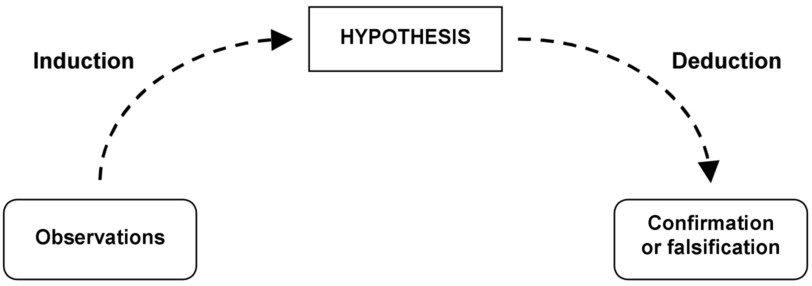


Figure 3.1: Inductive and deductive approaches to research.

3.1.1.3 The qualitative-quantitative dimension

Closely linked to the theory-building-and-testing dimension of ESE research, a distinction can also be made between qualitative and quantitative research. For the most part, *qualitative research* consists of studies that cannot be meaningfully quantified. These studies are typically in-depth analyses of one or a few observations, involving unstructured questioning or observation of the respondents. Overviews of the many facets of qualitative research can be found in (Denzin and Lincoln, 2000).

Quantitative research, on the other hand, typically uses larger samples and involves structured questioning or observations, which is subsequently numerically and statistically analyzed. A noted "bible" of quantitative research is (Cook and Campbell, 1979), while approaches that are more applicable for the general business situation can be found in (Cooper and Schindler, 1998; Davis, 1996).

Although there is often a heated debate between those who do qualitative research and those who do quantitative research (Reichardt and Rallis, 1994), it is our contention that ESE researchers should seek *interplay* between qualitative and quantitative methods. These issues are discussed in more detail in *Section 6.1* as part of the justification for and the assumptions behind the research approach used in this study.

3.1.1.4 The time dimension

Another dimension of ESE research is the treatment of time, which basically takes two forms (Davis, 1996):

- **Cross-sectional research**. In cross-sectional research designs, the measurements on the variables of interest are taken at one point in time. In essence, such designs provide the researcher with a snapshot of variables at one instant in time. Cross-sectional designs limit causal inferences because the study is conducted at one point in time and temporal priority is difficult to obtain.
- **Longitudinal research**. In longitudinal research designs, measures of the same sample or population are taken repeatedly (at least twice) through time. Typical longitudinal research types include time-series analysis, panel studies, and cohort studies. Panel studies involve the collection of data from the same respondents over a period of time. Cohort studies involve the collection of data about the same specific population over time, but a new sample is drawn from the population at every data collection point.

3.1.2 Fundamentals of research design

The primary purpose of a research design is to guide the researchers in their quest to answer the research question and solve the problems under study. This is accomplished through the careful construction of a research design so that the results obtained are as free of errors as possible. This is a complex undertaking, because of the numerous sources of potential errors that may affect the results of any investigation (see *Section 3.1.3*).

3.1.2.1 Types of variables

The variables in ESE research can be classified into one of four categories depending on their presumed relation to the phenomenon under study (Davis, 1996):

- **Independent variable**. The independent variable in a study is the presumed cause of the presumed effect. The independent variable is "independent of" prior causes that act on it. It produces a change in the dependent variable and is the one (or more, in the case of multivariate models) that the researcher believes precedes and affects the dependent variable.
- **Dependent variable**. The dependent variable in a study is the variable that is the effect or outcome of the independent variable. The dependent variable "depends on" the cause. In other words, the changes in the dependent variable are what the researcher try to predict, understand, or explain by using the independent variable.
- **Moderating variable**. A moderating variable is one that has a strong effect on an independent-dependent relationship. An example here is, as we discussed in the previous chapter, the influence of context on the process-outcome relationship in SPI (see *Figure 2.2*).
- **Intervening variable**. An intervening variable is one that emerges as a function of the independent variable operating in a situation and helps to explain the influence of the independent variable on the dependent variable.

3.1.2.2 Types of designs

There are several ways to classify ESE research designs. Broadly speaking, they can be divided into ex post facto and experimental designs (Campbell and Stanley, 1963). The distinction between these types is largely concerned with the researchers' control over the independent variables chosen in the study.

- **Ex post facto designs**. In ex post facto designs, the researcher does not attempt to manipulate the independent variable, because it is not manipulable for some reason or another.
- **Experimental designs**. In experimental designs, the researcher manipulates or in some way controls the independent variable and then measures the effect on the dependent variable of interest.

As we shall see in *Section 3.2.1*, experimental designs can be further divided into several sub-groups.

3.1.3 *General threats to validity*

Validity has to do with limiting research error in order to produce accurate and useable results. A fundamental question concerning empirical research, therefore, is how valid the results are. There are different classification schemes for different types of threats to validity in empirical research. Campbell and Stanley (1963), for example, defined two types: threats to internal validity and threats to external validity. Cook and Campbell (1979) extended the list to four threats to validity: statistical conclusion validity, internal validity, construct validity, and external validity.

The specific threats to validity for the different research approaches are discussed later in this chapter, in the sections describing each particular approach. Here, however, we give an overview of the most basic types based on Campbell and Stanley's (1963) classification.

3.1.3.1 Threats to internal validity

Internal validity is the *sine qua non* of a research design, and can be defined as the degree of confidence that the results are true given the study situation. Without internal validity there can be no confidence that the relationships identified in the investigation are really justifiable. A difficulty, however, comes in assessing internal validity, because it is never fully measurable. There are simply too many sources of invalidity to measure, or even identify, in the software engineering research environment. The major classes of variables that may affect a study's internal validity include (Campbell and Stanley, 1963):

- **History**, the specific events occurring between the first and second measurement in addition to the experimental variable.
- **Maturation**, processes within the respondents operating as a function of the passage of time per se, including growing older, growing hungrier, growing more tired, and the like.
- **Testing**, the effects of taking a test upon the scores of a second test.
- **Instrumentation**, in which changes in the calibration of a measuring instrument or changes in the observers or scorers used may produce changes in the obtained measurements.
- **Statistical regression**, operating where groups have been selected on the basis of their extreme scores.
- **Selection**, biases resulting in differential selection of respondents for the comparison groups.
- **Mortality**, or differential loss of respondents from the comparison groups.
- **Selection-maturation interaction**, etc., which in certain experimental designs might be mistaken for the effect of the experimental variable.

These are by no means all the threats to internal validity (see e.g. Cook and Campbell, 1979), but they do represent the major classes of variables that have been known to affect a study's results.

3.1.3.2 Threats to external validity

External validity can be defined as the degree to which the study's results can be generalized across populations, settings, and other similar conditions. However, a study cannot have external validity without evidence of internal validity. The major classes of variables that may affect a study's external validity include (Campbell and Stanley, 1963):

- **Testing interaction**, in which a pretest might increase or decrease the respondent's sensitivity or responsiveness to the experimental variable, making the results obtained for a pretested population unrepresentative of the effects of the experimental variable.
- **Selection interaction**, in which the effect that the type of respondents has on a study's results may limit its generalizability.
- **Setting interaction**, in which the artificial effects that are created by the specific setting of the study may not be replicated in other situations.
- **Multiple-treatment interference**, which is likely to occur whenever multiple treatments are applied to the same respondents, because the effects of prior treatments are not usually erasable.

3.1.4 Guidelines for increasing validity

In an attempt to increase the validity of empirical software engineering research, Kitchenham *et al*. (forthcoming) have offered a set of preliminary guidelines that can be used to improve the quality of on-going and proposed empirical studies and to encourage critical assessment of existing studies (see *Table 10.1*).

Furthermore, general guidelines for conducting and presenting empirical software engineering can be found in (Fenton, 2001) and for applying software engineering research results in (Pfleeger, 1997). A set of principles for conducting and evaluating interpretive field studies in information systems (IS) is presented in (Klein and Myers, 1999), while specific recommendations for increasing the relevance of IS research can be found in (Benbasat and Zmud, 1999) (see *Table 10.2*). Finally, guidelines for reporting experimental software engineering results based on the American Psychological Association's (APA) Publication Manual can be found in (Singer, 1999).

The adoption of such guidelines will not only improve the quality of individual studies, it will also increase the likelihood that we can use meta-analysis to combine the results of related studies.

3.2 Empirical Research Approaches

Three methods of investigation tend to dominate ESE research: formal (or laboratory) experiments, case studies, and surveys (Fenton and Pfleeger, 1996; Kitchenham, 1996/1998; Pfleeger, 1994/1995; Wohlin *et al*., 2000). In this section, we describe the features and applicability of each of these approaches. In addition, we describe two additional approaches that we consider of specific relevance for SPI: field experiments and action research.

3.2.1 Experimental research

An experiment is an empirical inquiry that investigates explanatory relations. It is especially well suited for answering questions about how and why (Yin, 1994). Experiments are launched when the investigator wants control over the situation, with direct, precise, and systematic manipulation of behavior (Wohlin *et al.*, 2000).

All experiments involve at least a treatment, an outcome measure, units of assignment, and some comparison from which change can be inferred and (hopefully) attributed to the treatment. *Randomized (or true) experiments* are characterized by the use of initial random assignment for inferring treatment-cause change. *Quasi-experiments*, on the other hand, also have treatments, outcome measures, and experimental units, but they do not use random assignment to create the comparisons from which treatment-caused change is inferred. Instead, the comparisons depend on nonequivalent groups that differ from each other in many ways other than the presence of a treatment whose effects are being tested. The task of interpreting the results from a quasi-experiment is, thus, basically one of separating the effects of a treatment from those due to the initial non-comparability between the average units in each treatment group, since only the effects of the treatment are of research interest (Cook and Campbell, 1979).

Experimental research is known as the standard method for empirical study in sciences such as physics, chemistry, and biology, but it is also an important part of disciplines such as medicine and psychology (see e.g. Anastasia and Urbina, 1997).

The need for experimentation in software engineering was first emphasized by Basili *et al.* (1986). Victor R. Basili has since then been one of the most prominent experimentalists in software engineering (see e.g. Basili, 1993, 1996; Basili and Selby, 1991). Others have also argued for an experimental approach to software engineering (e.g. Tichy, 1998; Zelkowitz and Wallace, 1998).

While experiments can help with induction, their most important application is in testing theories and hypotheses according to the hypothetico-deductive approach. In ESE, experiments are typically used to confirm the claims of theories or "conventional wisdom", to explore relationships among data points describing one variable or across multiple variables, to evaluate the accuracy of models, or to validate measures (Fenton and Pfleeger, 1996). Since experiments must be carefully controlled, they are often small in scale: "research-in-the-small" (Kitchenham *et al.*, 1995).

General guidelines for experimental design and analysis can be found in (Campbell and Stanley, 1963; Cook and Campbell, 1979). An introduction to statistics for experimenters is provided by (Box *et al.*, 1978). Specific guidelines for conducting software engineering experiments can be found in (Basili *et al.*, 1986; Fenton and Pfleeger, 1996; Pfleeger, 1994/1995). Furthermore, Wohlin *et al.* (2000) have written a pragmatic and short introductory textbook on experimentation in software engineering. Overviews of published software engineering experiments and their results can be found in (Basili *et al.*, 1986; Zendler, 2001).

3.2.1.1 Experimental models

In contrast to the *analytic paradigm*, which Basili (1993) illustrated by the mathematical method of model manipulation, he discussed three experimental models, or versions of the *experimental paradigm* in software engineering:

- **The scientific method**, which is an approach to model building; an *inductive paradigm* that might best be used when trying to understand the software process, product, people, or environment. It attempts to extract from the world some form of model which tries to explain the underlying phenomena, and, by measurement and analysis, evaluate whether the model is representative for the phenomenon being observed.
- **The engineering method**, which is also referred to as the *evolutionary paradigm*, is an improvement-oriented approach that assumes that we already have models of the software process, product, people, and environment. Based on observations of these models, better solutions are suggested, developed, and tested in an iterative manner until no more improvements seem possible. This is similar to the *improvement approach* described in the previous chapter.
- **The empirical method**, which is also referred to as the *revolutionary paradigm*, is an improvement-oriented approach that begins by proposing a new model, not necessarily based upon an existing model, and attempts to study the effects of the process or product suggested by the model. Similar to the *innovation approach* described in the previous chapter, observations or current solutions are not the basis for the model proposal. Wohlin *et al.*'s (2000) model of the experimental process, which builds on Basili *et al.*'s (1986) framework for experimentation in software engineering, is an example of the empirical method.

Common to these models is that they are guided by some rational for collecting data on either the software process or the resulting product. That is, they require an experimental design, observation, data collection, and validation on the process or product being studied.

3.2.1.2 Experimental designs

There are three general principles that guide experimental designs (Fenton and Pfleeger, 1996; Wohlin *et al.*, 2000):

- **Randomization**. Randomization is the random assignment of subjects to groups or of treatments to experimental units, so that independence (and thus validity) of results can be assumed. This is one of the most important experimental design principles, since all statistical methods used for analyzing the data require that the observations are from independent random variables. By randomly assigning treatments to experimental units, treatment results will be less likely to be biased by sources of variation outside the control of the experiment.
- **Blocking**. Blocking is the allocation of experimental units to blocks or groups so the units within a block are relatively homogeneous. The blocks are designed so that the predictable variation among units has been confounded with the effects of the blocks. That is, the experimental design captures the anticipated variation in the blocks by grouping like varieties, so that the variation does not contribute to the experimental error. In this way, we increase the precision of the experiment.
- **Balancing**. Balancing is the blocking and assignment of treatments so that an equal number of subjects is assigned to each treatment, wherever possible. Balancing is

desirable because it simplifies the statistical analysis of the data, but it is not necessary. Experimental designs can range from completely balanced to little or no balance.

In experiments investigating only one factor, blocking and balancing play important roles. If the design includes no blocks, then it must be completely randomized. If one blocking factor is used, subjects are divided into blocks and then randomly assigned to each treatment. In such *randomized block designs*, balancing is essential for analysis, and they are therefore often referred to as *complete balanced block designs*. If units are blocked with respect to two different variables and then assigned at random to treatments so that each blocking variable combination is assigned to each treatment an equal number of times, then balancing is mandatory for correct analysis. Such designs are called *Latin Square designs*.

However, all experimental designs can be seen as variations of the classic, *pretest-posttest control group design*. This design has random assignment, a pretest and a posttest, an experimental group, and a control group. Using Campbell and Stanley's (1963) notational system, the design can be represented as:

$$\begin{array}{llll} R & O_1 & X & O_2 \\ R & O_3 & & O_4 \end{array} \qquad (3.1)$$

where O refers to a measurement or observation of the dependent variable being taken on some individual, group, or object. X represents the exposure of a test group to an experimental treatment (independent variable), such as the introduction of a new method, tool, or technique. R means that individuals or groups have been selected and assigned at random for the study's purposes.

A simplified version of the pretest-posttest control group design, is the *posttest-only control group design*, which can be represented as:

$$\begin{array}{lll} R & X & O_1 \\ R & & O_2 \end{array} \qquad (3.2)$$

This design is useful in situations in which pretests are unavailable, inconvenient, or likely to be reactive. The primary strength of this design is ensured by the randomization of groups. A third, but less used experimental design is the *Solomon four-group design*, which is a combination of design (*3.1*) and (*3.2*) described above.

3.2.1.3 Types of experiments

Several taxonomies have been proposed to classify software experiments. Basili *et al.* (1986), for example, characterized experiments by the number of different projects analyzed and the number of teams replicating each project. This classification resulted in four classes of experiments: single project, replicated project, multi-project variation, and blocked subject-project. Blocked subject-project and replicated project experiments represent what Campbell and Stanley (1963) called true experimental designs, while multi-project variation and single project experiments represent quasi-experimental or pre-experimental designs.

Kitchenham (1996/1998) differentiated between qualitative and quantitative experiments, while Zelkowitz and Wallace (1998) defined four types of experiments: replicated experiment, synthetic environment experiments, dynamic analysis, and simulation.

For our purposes, however, we find it more useful to differentiate between the degree of realism in the research setting (Basili, 1996; Galliers, 1992):

- **Laboratory experiments** – also called *in vitro* experiments. The key feature of laboratory experiments is the identification of the precise relationships between variables in an isolated, controlled setting using quantitative analytical techniques. The idea is to make generalizable statements from the laboratory applicable to real world situations.

  The primary strength of the laboratory experiment is that the researcher has almost complete control of the study situation. However, the artificiality of the research setting may actually generate changes in the dependent variable that might not happen in the real world. Also, the artificiality of the experimental process may actually change the behavior of the subjects in the experiment (Davis, 1996). A famous example is the Hawthorne Studies, where it has been argued that it was the experimentation procedures themselves that changed the behavior of the workers (Carey, 1967).
- **Field experiments** – also called *in vivo* experiments. Field experiments are an extension of laboratory experiments into the real world of organizations; they are run in the field under normal conditions. The idea is to construct an experiment in a more realistic environment than is possible in the artificial, sanitized laboratory situation.

  The primary strengths of a field experiment are that (1) the study situation is usually highly realistic, allowing the effects of the independent variables to be accurately assessed, and (2) compared with ex post facto research, a stronger inference can be made about the relationship between the variables under study. However, the realistic environment can also be a weakness, because it may be too costly or impossible to manipulate an independent variable or to randomize treatments in real life.

The European Commission sponsored more than 300 SPI field experiments through the European Systems and Software Initiative (ESSI) during the 1990s, which is probably the single largest SPI effort undertaken anywhere. A review of results from these experiments can be found in (Consolini and Fonade, 1997), while a library of actual results is contained in the VASIE Library (see www.esi.es/VASIE).

3.2.1.4 Threats to experimental validity

Four criteria are commonly used to judge the quality of experimental designs (Cook and Campbell, 1979):

- **Statistical conclusion validity**, which is concerned with issues that affect the ability to draw the correct statistical conclusion about relationships between the treatment and the outcome of an experiment. Threats to statistical conclusion validity include low statistical power, violated assumptions of statistical tests, the error rate problem, the reliability of measures, the reliability of treatment implementation, random irrelevancies in the experimental setting, and random heterogeneity of respondents (see (1) in *Figure 3.2*).

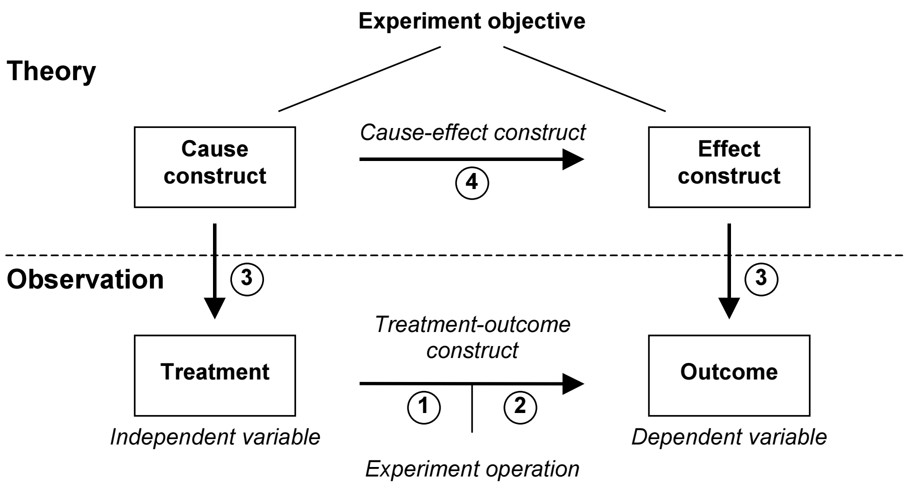


Figure 3.2: Threats to experimental validity (Wohlin *et al.*, 2000, p. 64).

- **Internal validity**, which is concerned with influences that can affect the independent variable with respect to causality, without the researcher's knowledge. The internal validity threats are sometimes sorted into three categories: single group threats, multiple group threats, and social threats (see (2) in *Figure 3.2*).
- **Construct validity**, which is concerned with generalizing the results of the experiment to the concept or theory behind the experiment. Threats to construct validity include inadequate preoperational explication of constructs, mono-operation and mono-method bias, hypothesis guessing within experimental conditions, evaluation apprehension, experimenter expectancies, confounding constructs and levels of constructs, interaction of different treatments, interaction of testing and treatment, and restricted generalizability across constructs (see (3) in *Figure 3.2*).
- **External validity**, which is concerned with conditions that limit the researcher's ability to generalize the results of the experiment to industrial practice. Threats to external validity include interaction of selection and treatment, interaction of setting and treatment, and interaction of history and treatment (see (4) in *Figure 3.2*).

There are potential conflicts between some of the types of threats to validity in an experimental design. Prioritizing among them is, therefore, an optimization problem, given the purpose of the experiment. Compared with laboratory experiments, field experiments are generally weaker in terms of control of extraneous variables that can threaten validity. Also, the different subtypes of experimental designs focus on different threats to validity.

3.2.2 *Survey research*

A survey is a retrospective study of a situation that investigates relationships and outcomes. It is useful for studying a large number of variables using a large sample size and rigorous statistical analysis. By combining the advantages of experiments (replication that minimizes the problems of unusual results) with those of case studies (applicability to real-world projects), a survey is a particularly useful empirical strategy (Fenton and Pfleeger, 1996).

Surveys conducted for research purposes have three distinct characteristics (Pinsonneault and Kraemer, 1993). First, the purpose of the survey is to produce quantitative descriptions of some aspects of the studied population. Second, the main way of collecting data is by asking people structured and predefined questions. Third, data is generally collected about a fraction of a study population – a sample – but it is collected in such a way as to be able to generalize the findings to the population. Usually the sample is large enough to allow extensive statistical analysis.

Surveys are especially well suited for answering questions about what, how much, and how many as well as questions about how and why (Pinsonneault and Kraemer, 1993). They are used when control of the independent and dependent variables is not possible or not desirable, when the phenomena of interest must be studied in their natural setting, and when the phenomena of interest occur in current time or the recent past.

Survey research is a standard method of empirical study in disciplines such as marketing, medicine, psychology, and sociology. There is also a long tradition for the use of surveys as an intervention strategy for organizational change (e.g. Baumgartel, 1959; Neff, 1966; Kraut, 1996). In ESE, surveys usually poll a set of data from an event that has occurred to determine how the population reacted to a particular method, tool, or technique, or to determine trends or relationships. They try to capture what is happening broadly over large groups of projects: “research-in-the-large” (Kitchenham *et al.*, 1995). The most common forms of ESE surveys are based on distributing questionnaires to elicit opinions about the benefits of technology, or to assess the effects of process changes throughout an organization.

Together with experimental research, survey research is a traditional “hard-science” approach that is supported by a rich literature describing how to design and administer it. A general review of survey research from the vantage point of psychology is provided by (Krosnick, 1999). General introductions and guidelines for survey research can be found in (Davis, 1996; Fink and Kosecoff, 1998; Neuman, 2000), while an assessment of survey research in management information systems can be found in (Pinsonneault and Kraemer, 1993). Details regarding instrument design and scale development is given in (Carmines and Zeller, 1979; DeVellis, 1991; Spector, 1992). An example of the construction of an instrument for ESE survey research can be found in *Chapter 7* and also in (Dybå, 2000a).

3.2.2.1 Survey designs

Survey designs may be distinguished as cross-sectional or longitudinal, depending upon whether they exclude or include explicit attention to the time dimension (Pinsonneault and Kraemer, 1993):

- **Cross-sectional design**. A cross-sectional design is used when the researcher’s aim is to describe a population or document and test differences in subsets of the population at one point in time. The classic cross-sectional survey design collects data at one point in time from a sample selected to represent the population of interest at that time.
- **Longitudinal design**. A longitudinal design is used when the question or problem of interest is the examination of a dynamic process that involves change over time and understanding of the sources and consequences of a phenomenon. The classic

longitudinal survey design collects data for at least two points in time. The underlying principle of longitudinal designs, like that of the *one-group pretest-posttest design* (Campbell and Stanley, 1963), is to measure some dimensions of interest of a given entity before and after an intervening phenomenon to determine whether or not the phenomenon has some effects.

Another critical issue in survey research design is determining the unit(s) of analysis. It may be an individual, group, department, or organization, or it may be an application, system, software project, or any of the processes of a software project. There can also be more than one unit of analysis in a survey. The point is that the chosen unit relates to the questions and hypotheses in the research.

Yet another issue is data collection, which can be either passive or active. *Passive data collection* involves the observation of characteristics, by human or computerized means, of the elements under study. *Active data collection*, on the other hand, involves the querying of respondents, by personal or nonpersonal means, using such methods as personal interviewing or mailed questionnaires.

A final issue is data analysis. When exploration or description is the aim of the survey, analysis frequently involves no more than developing the marginal and cross-tabulations for the variables, using simple descriptive statistics. When explanation is the aim, analysis includes the testing of hypotheses with cross-sectional data. This requires that the researcher designs the survey to include data on the independent and dependent variables and on such antecedent variables as theory would suggest might explain the expected original relation.

3.2.2.2 Sampling design

Sampling is concerned with drawing individuals or entities from a population in such a way as to permit generalization about the phenomena of interest from the sample to the population. A good sample has the following general characteristics (Davis, 1996):

- It enables the researcher to make decisions concerning what sample size to take to obtain the answers desired.
- It identifies the chance, or probability, that any primary unit of analysis will be included in the final study sample.
- It enables the researchers to quantify the accuracy and imprecision (errors) in choosing a sample, rather than taking a complete canvas of the population (a census).
- It enables the researchers to quantify the degree of confidence that can be placed in population estimates made from sample statistics.

These characteristics are applicable only for *probability designs*, which are designs in which each element in the population has a known, nonzero chance of being selected for inclusion in the study sample. In *nonprobability designs*, on the other hand, the chance of each element being selected is not known.

The most critical element of the sampling process is the selection of the sampling frame (Davis, 1996), because if the chosen sampling frame does not adequately represent the unit of analysis, the generalizability of the results of the study will be questionable.

3.2.2.3 Instrument design

Instrument design is very much an art (Payne, 1951). Difficulties do not only include the phrasing of questions, but also the determination of what to include and exclude. Formally, *we can define instrument design as the process of developing the data collection device (usually a questionnaire) in order to define and obtain relevant data for a given research question* (Dybå, 2000a).

There are many reasons why researchers in SPI should pay closer attention to instrumentation. First, concerns about instrumentation are closely connected with concerns about rigor in ESE research. Second, greater attention to instrumentation permits confirmatory, follow-up research to use tested instruments, hence, promoting cooperative research efforts (Hunter and Schmidt, 1990). Third, closer attention to instrumentation brings greater clarity to the formulation and interpretation of research questions (Straub, 1989). Finally, lack of validated measures in confirmatory research raises serious questions about the trustworthiness of the findings of the study.

Measurement of research constructs is neither simple nor straightforward. However, instrumentation techniques are available that allow us to construct research instruments that constitutes acceptable levels of reliability and validity. An example of a process for developing a research instrument for SPI based on generally accepted psychometric principles can be found in (Dybå, 2000a) and also in *Chapter 7* of this thesis.

3.2.2.4 Threats to survey validity

Four criteria are commonly used to judge the quality of survey research designs (Carmines and Zeller, 1979; DeVellis, 1991; Spector, 1992):

- **Reliability**, which refers to the consistency and stability of a score from a measurement scale. Basically, four methods are available for estimating the reliability of survey instruments: the test-retest method, the alternative form method, the split-halves method, and the internal consistency method.
- **Content validity**, which refers to the degree to which the items in the measurement instrument represent the domain or universe of the processes under study. Content validity is built into the instrument from the outset through the choice of appropriate items, which can be achieved by literature reviews, exploratory studies, expert reviews, and pilot tests.
- **Construct validity**, which refers to the degree to which the measurement instrument represents and acts like the processes being measured. Construct validity of the scales in a measurement instrument is normally assessed with factor analysis and item analysis based on Nunnally's method (Nunnally, 1978; Nunnally and Bernstein, 1994).
- **Criterion-related validity**, which refers to the degree to which the measurement instrument is able to predict a variable that is designated a criterion. The criterion-related validity of a measurement instrument can, thus, be found by assessing the effect size and significance level of the (multiple) correlation between the independent and dependent variables.

3.2.3 *Case study research*

A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident (Yin, 1994). So, while an experiment deliberately divorces a phenomenon from its context and a survey's ability to investigate the context is limited, the case study deliberately aims at covering the contextual conditions.

Generally, the most important application of case studies is to *explain* the causal links in real-life interventions that are too complex for the survey or experimental approaches. Another application is to *describe* an intervention and the real-life context in which it occurred. Furthermore, case studies can be used to *illustrate* certain topics within an evaluation or to *explore* those situations in which the intervention being evaluated has no clear, single set of outcomes. Finally, a case study may be a "*meta-evaluation*" – a study of an evaluation study (*ibid.*).

Case study research is a standard method of empirical study in management, but also in related disciplines such as organization development and information systems research. In ESE, case studies usually look at what is happening on a typical project: "research-in-the-typical" (Kitchenham *et al.*, 1995).

Yin (1994) noted that a case study has a distinct advantage when "a 'how' or 'why' question is being asked about a contemporary set of events over which the investigator has little or no control." (*ibid.*, p. 9). For software engineering, case studies are also useful in answering a "which is better" question (Kitchenham *et al.*, 1995).

Standard texts on case study research include (Stake, 1995; Yin, 1994). A discussion of the case research strategy in studies of information systems can be found in (Benbasat *et al.*, 1987). A scientific methodology for management information system (MIS) case studies is provided by (Lee, 1989), while the process of building theories from case study research is covered by (Eisenhardt, 1989). Guidelines and checklists for case studies in software engineering can be found in (Glass, 1997; Kitchenham, 1996/1998; Kitchenham *et al.*, 1995).

3.2.3.1 Case study designs

In general, case study designs can be single-case or multiple-case studies, and they can involve a single unit (holistic) or multiple units (embedded) of analysis (Yin, 1994). There are, thus, four general designs for case studies: (1) single-case, holistic design, (2) single-case, embedded design, (3) multiple-case, holistic design, and (4) multiple-case, embedded design.

In software engineering, case studies are particularly important for industrial evaluation of software engineering methods and tools because they can avoid the scale up problems that often are associated with experiments. To avoid bias and ensure internal validity, it is necessary to identify a valid basis for assessing the results of the case study. Basically, there are three ways of designing a software engineering case study to facilitate this (Kitchenham *et al.*, 1995):

- **Company baseline case studies**. This design can be used if the software organization gathers data from its projects as a standard practice and makes them available to the rest of the organization. With this design, the response variable values from the case study project are compared with the corresponding variables from previous projects or a subset of similar projects.

- **Sister project case studies**. A sister project design comprises (at least) two projects: one using the new method, and the other using the current method. Each project should be typical for the organization, having similar characteristics according to the chosen state variables. The projects are run in parallel and the response variables from each are compared.

  A special kind of sister project design is the *replicated product design*, which can be used in situations in which a product is developed a second time using a different development method.
- **Within project component comparison case studies**. In this design, components (e.g. modules or subsystems) in a single project are assigned at random to each of the treatments. In this case, the case study resembles a formal experiment, since the replicated values and standard statistical methods can be used to analyze the response variables. However, since the projects are not drawn at random from the population of all projects, this design is not a true, formal experiment.

3.2.3.2 Threats to case study validity

Four criteria are commonly used to judge the quality of case study research designs (Yin, 1994):

- **Construct validity**, which is concerned with establishing correct operational measures for the concepts being studied. Critics of case study research often point to insufficiently operational sets of measures and to the subjective judgements often used to collect the data. However, three tactics are available to increase construct validity in case study research: using multiple sources of evidence, establishing a chain of evidence, and having the draft case study report reviewed by key informants.
- **Internal validity**, which is concerned with establishing causal relationships as distinguished from spurious relationships, thus minimizing the effects of confounding factors. The requirements for internal validity, which are only applicable for explanatory case studies, can be met by the use of pattern-matching, explanation-building, and time-series analysis.
- **External validity**, which is concerned with establishing the domain to which a study's findings can be generalized. Critics typically state that single cases offer a poor basis for generalizing. In contrast to survey research that relies on statistical generalization, case study research relies on analytical generalization using replication logic in multiple case studies.
- **Reliability**, which is concerned with demonstrating that the operations of the study can be repeated, with the same results. For case study research, the emphasis here is on doing the *same* case over again, not on "replicating" the results of one case by doing *another* case study. The goal of reliability is, thus, to minimize the errors and biases in a study, which can be achieved by the use of a case study protocol and a case study database.

In discussing bounded rationality and organizational learning, the 1978 winner of the Nobel Prize in Economics, Herbert A. Simon made the following comment regarding the validity of case study data:

> If we are concerned about the imprecision of case studies as research data, we can console ourselves by noting that a man named Darwin was able to write a very persuasive (perhaps even correct) book on the origin of species on the basis of a study of the Galapagos Islands and a few other cases. To the best of my recollection, there are no statistics in Darwin's book (Simon, 1991).

3.2.4 Action research

Action research is particularly focused on combining theory and practice (Greenwood and Levin, 1998, 2000). It attempts to achieve practical value to the client organization while simultaneously contributing to the generation of new theoretical knowledge (Galliers, 1992). It can be characterized as "an iterative process involving *researchers and practitioners acting together* on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning." (Avison *et al.*, 1999, p. 94, emphasis added). The major strength of action research is, thus, the in-depth and first hand understanding the researcher obtains. On the other hand, the weakness is the potential lack of objectivity stemming from the researchers in effecting a successful outcome for the client organization (Benbasat *et al.*, 1987).

General introductions to action research and its varieties can be found in (Elden and Chisholm, 1993; Greenwood and Levin, 1998). A general discussion on the applicability of action research to IS research is provided by (Avison *et al.*, 1999; Mathiassen, 1998), specific frameworks for action research in IS is presented by (Baskerville and Wood-Harper, 1998; Lau, 1999), while a critical perspective on action research as a method for IS research can be found in (Baskerville and Wood-Harper, 1996).

In our view, action research is a highly underrated strategy for ESE research. In contrast to the other empirical approaches described in this chapter, the action researcher is not a neutral observer, detached from practice. Rather, the action researcher takes active part in the organization's change process. This is also an important part of the industry's motivation to participate in ESE research projects – that the researchers actively participate with relevant knowledge and experience to help the organizations improve themselves (see *Chapter 9*).

Also, unlike the other empirical approaches described in this chapter, which for the most part, rests in the *positivist* paradigm, action research rests in an *interpretive* philosophical framework (Galliers, 1992). In *Chapter 6*, we discuss the relationship between this framework and the positivist framework, and how they are mixed to form a pragmatic research design for this study. At this point, however, it could be useful to describe the basic principles for interpretive field research.

3.2.4.1 Principles for interpretive field research

In discussing the conduct and evaluation of interpretive research in IS, Klein and Myers (1999) proposed a set of seven principles along with their philosophical rationale (see *Table 3.2*). Furthermore, they illustrated the usefulness of these principles by evaluating three published interpretive field studies drawn from the IS research literature.

Table 3.2: Principles for interpretive field research (Klein and Myers, 1999).

1. The Fundamental Principle of the Hermeneutic Circle This principle suggests that all human understanding is achieved by iterating between considering the interdependent meaning of parts and the whole that they form. This principle of human understanding is fundamental to all other principles.
2. The Principle of Contextualization Requires critical reflection of the social and historical background of the research setting, so that the intended audience can see how the current situation under investigation emerged.
3. The Principle of Interaction Between the Researchers and the Subjects Requires critical reflection on how the research materials (or "data") were socially constructed through the interaction between the researchers and participants.
4. The Principle of Abstraction and Generalization Requires relating the idiographic details revealed by the data interpretation through the application of principles one and two to theoretical, general concepts that describe the nature of human understanding and social action.
5. The Principle of Dialogical Reasoning Requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and actual findings ("the story which the data tell") with subsequent cycles of revision.
6. The Principle of Multiple Interpretations Requires sensitivity to possible differences in interpretations among the participants as are typically expressed in multiple narratives or stories of the same sequence of events under study. Similar to multiple witness accounts even if all tell it as they saw it.
7. The Principle of Suspicion Requires sensitivity to possible "biases" and systematic "distortions" in the narratives collected from the participants.

3.2.4.2 Action research models

Three distinct types of models can be seen in action research (Baskerville and Wood-Harper, 1998):

- **Iterative models**, which involve a repeating sequence of activities, typically cycling between action activities and problem diagnosis activities (e.g. Checkland and Scholes, 1990, 1999; Susman, 1983).
- **Reflective models**, which are necessarily iterative, but focus more on reflective analysis of theory-in-use *versus* espoused-theory (see *Section 4.4.1* for an overview). A reflective process model, thus, concentrates on the discovery of differences between the two theories (e.g. Greenwood and Levin, 1998; Naur, 1983).
- **Linear models**, which do not involve iteration, but a single sequence of activities such as engage, diagnose, unfreeze, change, freeze, and disengage (e.g. Mumford, 1983).

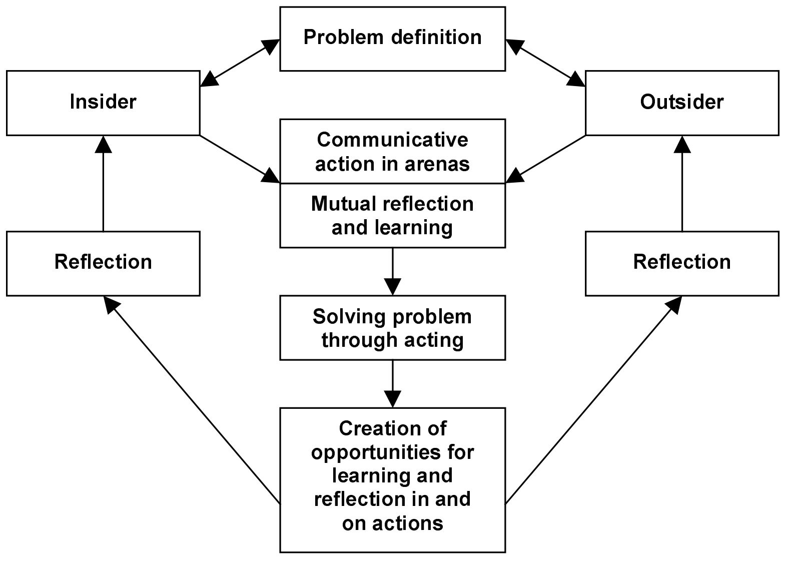

Problem definition
Insider
Outsider
Communicative action in arenas
Mutual reflection and learning
Reflection
Reflection
Solving problem through acting
Creation of opportunities for learning and reflection in and on actions

Figure 3.3: The Cogenerative Action Research Model (Greenwood and Levin, 1998, p. 116).

As an example of an action research model, *Figure 3.3* shows Greenwood and Levin's (1998) Cogenerative Action Research Model. This is a reflective model, which identifies two main groups of actors: the insiders and the outsiders of the organization. The *insiders* are the focal point of the action research; they "own" the problem. The *outsiders*, on the other hand, are the professional researchers who seek to facilitate a colearning process aimed at solving local problems.

- **Problem definition**. The problem definition process is the first step in a mutual learning process between insiders and outsiders. A first working definition of the problem under study comes out of a discourse where knowledge held by insiders and outsiders cogenerates a new, mutual understanding through communication with each other.
- **Communicative action in arenas**. Central to the cogenerative process is its ability to create contexts, or arenas, that allow communicative actions to take place. Examples of such arenas include various forms of meetings, team building sessions, and search conferences.
- **Mutual reflection and learning**. The initial problem focus suggests a design for an arena for discourse. The subsequent communication produces understandings that help move toward problem solutions, creating new experiences to reflect on for both insiders and professional researchers.
- **Solving problem through acting**. The struggle to solve important local problems shapes the ground for new understandings, hence the double feedback loops in *Figure 3.3*. Actions taken to solve local problems come as a result of the cogenerative process, through which the participants learn new things about the problems they are facing. The outcome of this collective process of action and reflection support the creation of new shared understanding.

- **Reflection**. The feedback loops are similar for both insiders and outsiders, but the interests and effects can be quite different. For insiders, it might be central to improve their action-knowledge capabilities, whereas the outsiders may focus more on producing meaning (through publications) to the research community. Both of these reflective processes are then fed back into the communicative process, shaping the arenas for new dialogues aimed at either redefining the initial problem statement or improving the local problem-solving capacity.

3.2.4.3 Threats to action research validity

Rather than the traditional concepts of reliability and validity, action researchers argue that action research should be judged according to its *credibility*, which Greenwood and Levin (1998) defined as "the arguments and the processes necessary for having someone trust research results." (*ibid.*, p. 80). Two types of credibility can be distinguished:

- **Internal credibility**, which is concerned with the credibility of knowledge to the group generating it. This type of credibility is fundamentally important to action research because of the collaborative character of the research process. Members of communities or organizations are unlikely to accept as credible the "objective" theories of outsiders if they cannot recognize the connection to the local situation, or because local knowledge makes it clear that the frameworks are either too abstract or simply wrong for the specific context.
- **External credibility**, which is concerned with convincing someone who, did not participate in the inquiry, that the results are believable. This is a difficult challenge, however, since action research depends on the conjugation of reflection and action and the cogeneration of new knowledge in specific contexts.

While conventional research methods focus on validity through statistical or analytical generalization, action research believes that only knowledge generated from and tested in practice is credible. This involves (at least) three challenges (*ibid.*):

- **Workability**. This credibility challenge relates to the solution of the action research question under examination locally, and whether the actions taken during the research process result in a workable solution to the problem.
- **Sensemaking**. The second and complementary challenge is making sense out of the tangible results of the action research process, finding ways of integrating the outcome in a sensemaking process that creates new knowledge. Examples of such processes include Habermas's (1984) ideal speech situation and Gadamer's (1989) hermeneutics.
- **Transcontextual credibility**. Action research does not generalize through abstraction and the loss of history and context. Rather, meanings created in one context are examined for their credibility in another situation through a conscious reflection on similarities and differences between contextual features and historical factors. They are moved from the original context through a collaborative analysis of the situation where the knowledge might be applied. Based on historical and contextual analyses, judgements are made about the possibility of applying knowledge from one situation in another.

3.3 Comparison of Research Approaches

The diversity in ESE research approaches applicable to SPI is, as we have seen, considerable. This is useful, since the complexity and dynamics of the software engineering problems under study often require that multiple approaches be applied. This diversity is, unfortunately, combined with a number of institutionalized specializations with rather little exchange between them (Mathiassen, 1998). First, research concerned with SPI is divided into two research communities: *software engineering*, which is practiced within computer science and engineering departments, and *information systems development*, which is mainly practiced as part of information and management sciences. A notable exception is NTNU, where the two communities have been merged into one cooperative research department: Department of Computer and Information Science.

Second, two different research traditions have emerged: a *positivist*, which is primarily occupied with structured, quantitative approaches based on observations and an *interpretivist*, which uses less structured, more qualitative approaches based on interpretations and intervention. For the most part, ESE research has been concerned with positivist approaches such as laboratory experiments. This bias toward and idealization of positivist approaches is most visible through the choice of terminology. In fact, several of the leading ESE researchers seem to mix the term "empirical" with "experimental". Zelkowitz and Wallace (1998), for example, classified a number of research approaches as "experimental models", which in our view has nothing do to with experiments (e.g. project monitoring, assertion) or even with empirical research (e.g. literature review).

Finally, there are considerable differences in what is considered the primary outcome of research: one group is focusing primarily on concepts, methods, and tools *to support practice*, and another is striving to develop theories and frameworks to improve our *understanding of practice*.

Each of the related communities have developed strong and relevant positions, and one of the key challenges in advancing ESE research is to improve the dialogue between them.

Rather than arguing for the general supremacy of one research approach or search for a synthesis or common ground for the "best way" of doing research, we argue that the researcher must engage in a reflective process, actively examining the research problem and the environment in which it will be studied. Based on such reflection, ESE researchers should take responsibility for making intelligent choices about the approach or combination of approaches they adopt and the ends they serve in each specific situation.

From an academic point of view, the ultimate criterion for making such choices is scientific rigor, which is often linked to the reliability of the instruments for data collection and analysis used in the research, and to the internal and external validity of the research findings. However, in an applied discipline such as SPI, relevance is at least as important as rigor. In our opinion, therefore, it is necessary that the ESE research community values the diversity in available research approaches, actively trying to understand and improve them to better serve both researchers and practitioners.

As will become more evident throughout the thesis, we consider the topic of SPI as a socio-technical subject, not simply a technical one. It is because of this that we have undertaken the present investigation of the importance of organizational issues in enabling SPI. Consequently, one might call into question the appropriateness of the biased focus on positivist approaches in present ESE research. In our view, ESE researchers should cooperate with practitioners and undertake more research with a pragmatic philosophy, combining positivist and interpretive approaches in real-world situations, rather than emphasizing experimental elegance in the sanitized environment of the laboratory (see *Chapters 6* and *9*).

The key features of the ESE research approaches described in this chapter, which are summarized in *Table 3.3* together with their major strengths and weaknesses, can be seen as a step toward familiarizing ESE researchers with some of the approaches available for their research. Possible ways of combining these approaches in the process of building, extending, and testing theory is presented in *Figure 3.4*.

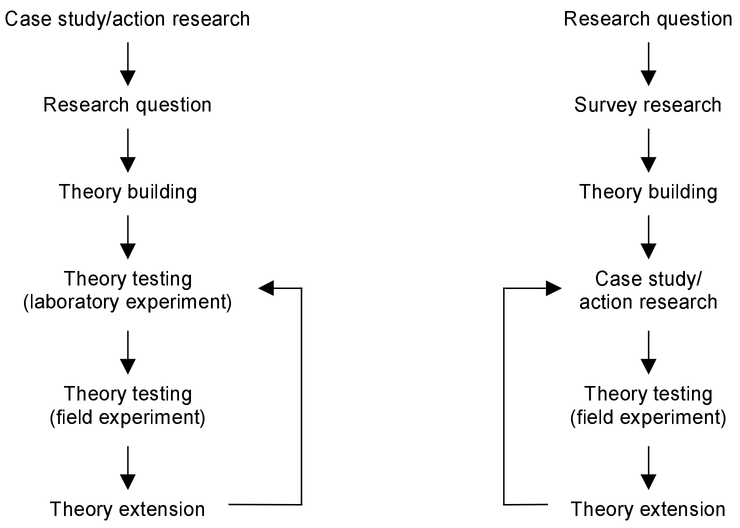

Case study/action research
Research question
Theory building
Theory testing (laboratory experiment)
Theory testing (field experiment)
Theory extension
Research question
Survey research
Theory building
Case study/ action research
Theory testing (field experiment)
Theory extension

Figure 3.4: The use of alternative ESE research approaches in the process of theory building, testing, and extension (Galliers, 1992, p. 161).

3.4 *Chapter Summary*

In this chapter, we have described the role of ESE research as one of encouraging a more scientific and reflective approach to software engineering, which allows us to investigate and understand new technologies and their applications to practice. Furthermore, four ESE research approaches that seeks to address this were identified and reviewed: experimental research, survey research, case study research, and action research. Finally, we made a comparison of the main features, strengths, and weaknesses of each of these approaches.

The next chapter introduces the model building phase of the present investigation by describing the underlying assumptions and foundations of the learning software organization, which form the basis upon which our dynamic model of SPI is constructed.

Table 3.3: A summary of the key features, strengths, and weaknesses of alternative ESE research approaches (adapted from Galliers, 1992, pp. 150-152).

Approach	**Key Features**	**Strengths**	**Weaknesses**
Laboratory experiments	Identification of precise relationships between chosen variables via a designed laboratory situation, using quantitative analytical techniques, with a view to making generalizable statements applicable to real-life situations.	The solution and control of a small number of variables which may then be studied intensively.	The limited extent to which identified relationships exists in the real world due to oversimplification of the experimental situation and the isolation of such situations from most of the variables that are found in the real world.
Field experiments	Extension of laboratory experiments into the real-life situation of organizations and/or society.	Greater realism; less artificial/sanitized than the laboratory situation.	Finding organizations prepared to be experimented on. Achieving sufficient control to enable replication, with only the study variables being altered.
Surveys	Obtaining snap shots of practices, situations or views at a particular point in time (via questionnaires or interviews) from which inferences are made (using quantitative analytical techniques) regarding the relationships that exist in the past, present and future.	Greater number of variables may be studied than in the case of experimental approaches. Description of real world situations. More easy/appropriate generalizations.	Likely that little insight obtained regarding the causes/processes behind the phenomena being studied. Possible bias in respondents (cf. self-selecting nature of questionnaire respondents), the researcher, and the moment in time which the research is undertaken.
Case studies	An attempt at describing the relationships which exist in reality, usually within a single organization or organizational grouping.	Capturing "reality" in greater detail and analyzing more variables than is possible using any of the above approaches.	Restriction to a single event/organization. Difficulty in generalizing, given problems of acquiring similar data from a statistically meaningful number of cases. Lack of control of variables. Different interpretations of events by individual researchers/ stakeholders.
Action research	Applied research where there is an attempt to obtain results of practical value to groups with whom the researcher is allied, while at the same time adding to theoretical knowledge.	Practical as well as theoretical outcomes most often aimed at emancipatory outcomes. Biases of researcher made known.	Similar to case study research, but additionally places a considerable responsibility on the researcher when objectives are at odds with other groupings. The ethics of the particular research are a key issue.

CHAPTER 4

Foundations of the Learning Software Organization

"*Reality is merely an illusion, albeit a very persistent one*."

– Albert Einstein

In this chapter, we describe the underlying assumptions and foundations of the learning software organization, which form the basis upon which our dynamic model of SPI is constructed (see *Chapter 5*). These foundations originate from our experience and observations, which led us to an emphasis on SPI as organizational learning – as a collective ability expressed in and through the organizations' product- and service-oriented practices related to software development. Furthermore, it led us to emphasize a more adaptive and active view of knowledge rather than the traditional static and passive view. The following premises are identified:

- **Social learning**, which is focused on SPI as a social, collaborative activity within the context of a learning software organization (see *Section 4.1*).
- **Sensemaking**, which is aimed at constructing meaning and expressing the basic assumptions and values that are vital to the software organization and its members (see *Section 4.2*).
- **Knowledge creation**, which is aimed at generating new knowledge and new competencies that enable or broaden the software organization's potential range of actions (see *Section 4.3*).
- **Purposeful action**, which is aimed at using the new interpretations and new knowledge to construct improved courses of action (see *Section 4.4*).

In the rest of this chapter we describe the theoretical foundations for each of these premises, which lie behind our dynamic model of the learning processes and the key factors for success in SPI. Finally, we summarize the chapter by linking these foundations to the dynamic model of SPI, which is defined in the next chapter.

4.1 Social Learning

Anyone engaged in constructing an environment for SPI is faced with the question of the proper kind of overall perspective to choose as a guideline. According to Morgan (1997), such perspectives can be created through the use of metaphors. He invites and challenges us to recognize and cope with the idea that all theories of organization and management are based on implicit images of metaphors that persuade us to see, understand, and imagine situations in partial ways. He argued that metaphors both create insight and distort at the same time by highlighting particular facets and obscuring others. Morgan's (1997) main argument was that once we have learned how to generate, integrate, and use the insights of competing metaphors, they could be used to understand and shape the situation that we are seeking to organize and manage.

In this section, we first take a closer look at the rationalistic perspective that dominates technology development today, and the consequences that this perspective has on SPI. Then we propose an alternative perspective, that software development and SPI can be seen as a social construction of reality, and argue that this is a necessary perspective for a meaningful conception of the learning software organization.

Rather than the narrow definitions of technology found within engineering, we adopt MacKenzie and Wajcman's (1985) three-level definition of technology as (1) physical objects or artifacts, (2) activities or processes, and (3) knowledge or potential actions. In this respect, both software, software development, and software process improvement can clearly be regarded as technology. Moreover, this definition also makes an important connection between technology and human activity.

4.1.1 The rationalistic tradition

Today, the dominant perspective on software development is rooted in the "best practice" paradigm, promoting a product-line approach to software development with a standardized, controllable, and predictable *software engineering* process. Within the "best practice" paradigm, we find that standardized routines and automation replace human reasoning, and that the whole process is aimed at an economical optimization of *re-production*.

The chief symbol of the rationalistic approach to management was Frederick Winslow Taylor, and his principles of scientific, or "task", management. Taylor's (1911) project was to maximize productivity in factory operations, by (1) externalizing the workers knowledge through observation and measurement, (2) separate planning from task execution, and (3) introducing specific control mechanisms. The idea was to have complete knowledge of each task, and to prepare detailed instructions about the work to be done *before* it was started. This, then, would free the workers from thinking about past experience – all they needed to do was to follow managerial directions. This view has also been forcefully claimed within software development (e.g. Roetzheim, 1988).

Furthermore, Max Weber observed the parallels between the mechanization and specialization of work in industrial settings and the proliferation of bureaucratic forms of organization. However, Weber (2000) also perceived bureaucracy as a power instrument, and he was concerned that the efficiency of bureaucratic organizations could all too easily lock us into an "Iron Cage" of machine-like existence.

March and Simon (1958, 1993) denied the possibility of finding an objective, theoretical "best way" in practice, and they attacked the unbound rational model of organizations. Consequently, they argued that the objective, practical goal was not to find the optimal solution, but to find one that is *good enough*. The central theme in March and Simon's (1958, 1993) work is the idea of limited, consequential rationality, which has become more or less standard in modern theories of decision making. Their main argument was that *because of the limits in human understanding and decision-making, organizations will never be fully rational or be able to adapt completely to their environments*.

Translating the notions of the rationalistic tradition into the domain of software development, yields the following set of assumptions (Winograd and Flores, 1986; Floyd, 1992):

- Software engineering is concerned with the production of software on the basis of fixed requirements that can be separated from its use and that can be obtained by analyzing the facts of a given reality.
- The essential task of software developers is to find a correct solution, in the form of software, to the problems defined in the models representing that reality.
- The production process is independent of individuals and, hence developers should be interchangeable. That is, different developers should be able to arrive at the same results.
- Subject to technical feasibility, any desired parts of the production process can be automated.

One of the great contributions of the rationalistic approach was the practice of clarifying and making the concepts of task management and organizational structure operational, so that managers could put it into action. Also, the view of software development reflected in these assumptions has been instrumental in bringing about impressive advances in software and software development methods. It allows us to understand important aspects of software development prior to initiation of the development process, to assess more or less completed projects, or to "fake" a rational design process as suggested by Parnas and Clements (1986).

Despite these advances, the rationalistic approach to software development has severe limitations. First, the fragmentation and authoritarian control systems are inflexible and can have great difficulty in adapting to changing circumstances. Second, it can have dehumanizing effects upon employees leading to group defense (Trist and Bamforth, 1951), alienation and loss of freedom, initiative, and creativity (Blauner, 1964). Third, the rationalistic approach remains a myth in two respects (Brunsson 1985): (1) it rests on simplifying assumptions that ignore the range of legitimizing functions which decisions can fulfil and (2) it presumes a causal link between ideas and action that is usually missing in practice. Fourth, the rationalistic approach fails to offer any help in understanding an actual software development process in a given situation. Finally, the "best practice" view of software development is neglecting the inherent social character of software development, and that the development environment is socially constructed through processes of communication in which individual developers define their situations.

Consequently, it follows from the above argumentation that *the realization of an objective "best way" to software development is not a practical possibility*. However, questioning the rationalistic approach necessarily involves examining alternative approaches. We have chosen an approach were we perceive software development as *reality construction*. To understand this position we must enter into the constructivist discourse.

4.1.2 Software development as reality construction

A software developer analyzing an organization in order to develop a software system to support its business processes, is, according to the rationalistic approach, normally encouraged to start from the "real world" to abstract and to elaborate a correct model that can be manipulated by the computer. While this may be difficult to do, the task itself – *discovering* the correct description – is supposed to be clearly defined and independent of the software developer as an individual. This picture changes drastically, however, when we acknowledge our active role in bringing about what we hold for real, which is the key to constructivist thinking.

Constructivism is a theory, which differs from the traditional view that knowledge exists independently of the individual – the view that the mind is a *tabula rasa*, a blank tablet upon which a picture can be painted. A fundamental premise of constructivism is that software developers actively construct their knowledge. Rather than simply absorbing ideas spoken at them by others, constructivism posits that software developers actually invent their ideas. They assimilate new information to simple, pre-existing notions, and modify their understanding in light of new data. In the process, their ideas gain in complexity and power, and with appropriate support they develop critical insight into how they think and what they know about the world as their understanding increases in depth and detail. Constructivism emphasizes the careful study of the processes by which individuals and groups create and develop their ideas. Its SPI applications lie in creating a learning environment that match – but also challenge – software developers' understanding, fostering further growth and improvement of their work.

Our particular interest has been to apply the social construction perspective within the context of software development and SPI. Several authors may be said to hold a similar view. Winograd and Flores (1986), for example, inquired into the processes of computer and software design using a constructivist approach. Similarly, Floyd *et al*. (1992) discussed how software development could be seen as reality construction, while Gjersvik (1993) studied the social construction of information systems in organizations. Also, Baetjer (1998), Bennetts *et al*. (1999), Krogstie (1995), Messnarz (1999a), and Siddiqi (1994) have relied on a constructivist approach in their software research. Furthermore, there is also a large body of literature regarding the social construction of technology in general (e.g. Bijker *et al*., 1987; Latour, 1987; MacKenzie and Wajcman, 1985).

Peter Berger and Thomas Luckmann's *The Social Construction of Reality* is probably the most influential source for recent constructivist ideas. Berger and Luckmann (1966) placed the sociology of knowledge, i.e. the study of the social conditions of knowledge, at the core of sociological theorizing: "The basic contentions of the argument of this book are ... that reality is socially constructed and that the sociology of knowledge must analyze the process in which this occurs." (p. 13).

According to Berger and Luckmann (1966), the construction of social reality is a dialectical process between *externalization* of individual subjective realities, through action, resulting in *objective reality*; and the *internalization* of this objective reality, through sensemaking, into *subjective reality*. In the context of software development, we find it more helpful, though, to use Gjersvik's (1993) terms "local" and "organizational reality" instead of Berger and Luckmann's (1966) concepts of "subjective" and "objective reality", as shown in *Figure 4.1*.

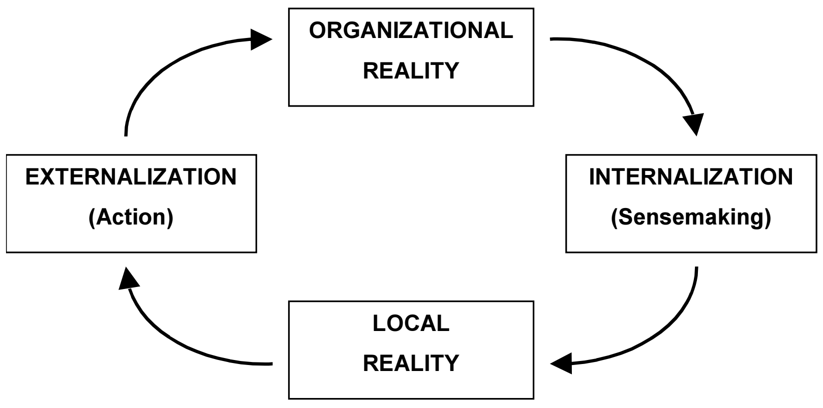


Figure 4.1: The construction of social reality (adapted from Gjersvik, 1993, p. 37).

Local reality is the way an individual or a group of individuals perceive the world in which they live. It is a kind of knowledge about how the different parts of the world fit together, what is related to what, and how to interpret actions and events (Gjersvik, 1993). Local reality is the knowledge that shows up in action, or what we might characterize as "knowing".

There are several social groups within a software organization that share knowledge and that may be identified as having a distinct local reality. Examples of such groups are formal project teams and informal groups of software developers and managers. The group's local reality can be seen as a way of acting in relationship to the rest of the organization. The strength of such local realities as with all socially constructed knowledge, comes from the "we-feeling" created by shared experience (Lysgaard, 1961).

This gives rise to the important point that, *since social groups define the problems of software development, there is flexibility in the way software is designed.* Hence, there is not just one possible way or one best way of designing a software product or defining its scope.

Externalization is the enactment of local reality. It is the process of creating organizational institutions, systems, and routines by taking action and using language to express oneself to the external world. It can be seen as a process of creating social order (Berger and Luckmann, 1966).

> [S]ocial order is a human product, or, more precisely, an ongoing human production. It is produced by man in the course of his ongoing externalization … Social order is not part of the 'nature of things', and it cannot be derived from the 'laws of nature'. Social order exists only as a product of human activity (*ibid.*, pp. 69-70).

Thus, through the process of externalization, organizational members construct organizational reality.

Organizational reality is the social order or institution that exists because everybody relates to it in their actions. It is, typically, perceived to have an existence apart from our interpretations and meanings. "Institutions generally manifest themselves in collectivities containing considerable numbers of people." (Berger and Luckmann, 1966, p. 73). Institutions both enable and limit what is possible to do depending on their history and degree of control.

> Institutions always have a history, of which they are the products. It is impossible to understand an institution adequately without an understanding of the historical process in which it was produced. Institutions also, by the very fact of their existence, control human conduct by setting up predefined patterns of conduct, which channel it in one direction as against the many other directions that would theoretically be possible (*ibid.*, p. 72).

Thus, the more conduct is institutionalized, the more predictable and thus the more controlled it becomes.

Internalization is the process by which individuals or groups of individuals give meaning to organizational reality, making it part of their own local reality. Whereas externalization can be seen as the way that the individual produces society, internalization can be seen as the way that the social world acts back upon its producer: "*Society is a human product … Man is a social product*." (Berger and Luckmann, 1966, p. 79, italics in original).

Berger and Luckmann (1966) considered two different phases of internalization: primary socialization and secondary socialization. Primary socialization is about how the child adopts his or her world while secondary socialization is about "the internalization of institutional or institution-based 'sub-worlds.'" (*ibid.*, p. 158). Within the context of this thesis, we are only concerned with the latter phase of internalization.

As can be seen from *Figure 4.1*, the social construction of reality is a simultaneous and continuous process of construction and reconstruction. It is a dialectic interplay between local and organizational reality, which is simultaneously characterized by externalization and internalization.

Constructivist thinking in software development leads to an emphasis on the *observer* constituting the way he or she sees reality and *inventing* a suitable description, rather than mapping a given reality. Thus, the software developer is portrayed as making *choices* in an open situation, where there is more than one possibility. When developing a software product, we make choices in selecting the aspects we consider relevant for modeling, in making available modes of interaction with the computer, in determining the software system's architecture, and in the way we use the technical resources for implementing the system. Moreover, we make choices in anticipating how the computer will be embedded in its use context and in creating facilities and constraints for users and other concerned parties. And finally, we make choices in how we conduct the development process itself, often deviating from pre-made plans.

Only a small part of these choices are made explicitly, more often they are implied by the course of action we take, or as Schön (1983) argued: "our knowing is *in* our action." (p. 49). Furthermore, each practitioner treats his or her case as unique, and consequently cannot deal with it by applying standard theories or techniques, hence, the practitioner "must construct an understanding of the situation as he finds it." (*ibid.*, p. 129). Also, our choices are constrained by our interactions with others. When seen in these terms, the task of software development clearly involves reference to the individual software developer.

Software development is, in accordance with the above argumentation, largely a human based intellectual activity (Weinberg, 1971, 1998) that involves constant negotiation and renegotiation within and between the social groups shaping the software. Software products emerge as the outcome of complex social processes, which serve to establish a common understanding of the problems to be dealt with and to anticipate meaningful solutions in the software's intended use context. Thus, we argue that *software development is a socially constructed process where the developers construct the process of development, the product, and the possibilities for its use through their collective choices and actions*. Furthermore, since a group of developers acquire the know-how necessary to carry out the collective task of software development, we contend that *organizational learning takes place as part of the practice*.

4.1.3 The learning software organization

So far we have argued for a perspective on software development as reality construction. In this section, we argue that this perspective requires a *commitment to learning* rather than to "best practice" models to accomplish improvements in software development processes. The basic premise for such improvements is that we expect to be able to *understand* why and how we do what we do in software development, so that we can do it deliberately and repeatedly in diverse and novel situations. Moreover, we want to be able to *externalize* our understanding of the development practice to be able to share it with others and to work with it directly to improve it.

Concordant to the rationalistic approach, the "best practice" perspective to SPI is to compare an organization's process with some generally accepted ideal or standard of best practices – *the one best way*. According to this perspective then, process improvement is the mere elimination of differences between the existing process and the ideal model. The assumption is that, once the process is changed to fit the prescribed model the generated products will be improved, or at least the quality risks of developing new software will be reduced (see *Chapter 2*).

Organizations are seen as learning by encoding inferences from history into routines that guide behavior (Levitt and March, 1988, p. 320).

An entity learns if, through its processing of information, the range of its potential behaviors is changed (Huber, 1991, p. 89)

Organizational learning means the process of improving actions through better knowledge and understanding. (Fiol and Lyles, 1985, p. 803).

Organizational learning is defined as increasing an organization's capacity to take effective action (Kim, 1993, p. 43).

Organizational learning occurs through shared insights, knowledge, and mental models ... [and] builds on past knowledge and experience – that is, on memory (Stata, 1989, p. 64).

We define organizational learning as a process in which an organization's members actively use data to guide behavior in such a way as to promote the ongoing adaptation of the organization (Edmonson and Moingeon, 1998, p. 12).

We define organizational learning as the capacity or processes within an organization to maintain or improve performance based on experience (Nevis *et al.*, 1995, p.73).

Figure 4.2: Alternative definitions of organizational learning.

As we argued in *Chapter 2*, and also in (Dybå and Skogstad, 1997), such model-based improvement can be contrasted with the use of improvement processes that are more concerned with the contingent characteristics of individual markets and organizations. Hence, we argue that process improvements can hardly be claimed unless process effectiveness is closely related to the specific circumstances, needs, and business goals of the organization. Furthermore, we concur with Nicolini and Meznar's (1995) view on organizational learning as a social construction, and argue that *software process improvement is a socially constructed learning process*. However, in order to understand how software organizations may become learning organizations we must first understand what organizational learning is.

Argyris and Schön (1978) introduced the concept of organizational learning in the 1970s. However, learning is basically an individualistic concept drawn directly from psychology, and, as can be seen from *Figure 4.2*, there is considerable disagreement among the most influential scholars concerning a good definition of what it means for an organization to learn.

Some authors, like Huber (1991), Kim (1993), and Nevis *et al*. (1995), claim that new ways of thinking are enough for an organization to learn. However, without accompanying change in the way that work gets done, only the *potential* for improvement exists. Argyris and Schön (1996) addressed this important aspect when they defined instrumental learning as "an organization's improvement of its task performance over time." (p. 4), and when they asserted that "change in behavior is a [necessary] condition for learning" (p. 33). Within the context of SPI, we hold that the role of organizational learning is to provide a framework for improved actions. Therefore, we adopt Fiol and Lyles' (1985, p. 803) problem-solving approach to organizational learning, and *define organizational learning within the context of SPI as the process of improving actions through better knowledge and understanding*.

The *learning software organization* offers an alternative to the pervasive factory view rooted in Weber's (2000) description of bureaucracy, Taylor's (1911) concept of scientific management, and the mechanistic notions of "best practice". Rather, the learning organization perspective is based on socio-technical principles (Trist, 1981; Trist and Bamforth, 1951) and participatory learning processes (Greenwood and Levin, 1998). The emphasis is on supporting *individual* activities by employing appropriate tools and techniques as the basis for cooperation between "multi-skilled" developers. What's more, the work of the software developers in the learning software organization can be characterized by teamwork and cooperation based on "optimum task grouping" and "minimum critical specification" (Trist, 1981). Of course, there is a division of labor and specialization, but still, there seems to be little chance of arriving at an overall state of routinized and mechanized work, based on established standards for software "best practice" and the "one best way".

The learning software organization can be characterized by its integration of intellectual work, coordination, and design. It is a collective of "semiautonomous groups" that brings planning, design, and construction activities together under the same roof, making it suitable for coping with individual requirements. Furthermore, its flexible forms of organization promote attunement to individual customer needs and wishes, securing "co-determination" – the direct participation of developers and users in decisions about what should best be done at their own level. The best practice approach, on the other hand, is suitable for reproductive working processes, geared at economical reproduction of a maximum number of goods that are specified in detail beforehand.

Since *planning*, *developing*, and *coordination* are the central tasks in software development – and not the mass production of goods – the learning software organization may be considered a proper metaphor for the working environment of software developers. The crucial aspect here is the close relationship between design, production, and evaluation: activities that bear the imprint of both individual and cooperative work.

To summarize, the "best practice" approach stands for replacing human labor by machinery and routinization. The learning software organization, on the other hand, promotes experience and the use of technology to enhance the skills of the developers rather than to deskill them (Corbett, 1992) – to "informate" rather than to "automate" tasks (Zuboff, 1988). *Table 4.1* summarizes the relationship between the "best practice" and the learning perspectives of software organizations.

Based on the insights of the preceding section, the rest of this chapter describes the remaining three foundations for the learning software organization that eventually shaped our dynamic model of SPI: *sensemaking*, *knowledge creation*, and *purposeful action*.

4.2 *Sensemaking*

In today's increasingly turbulent and unpredictable environments, software organizations have to make sense out of what is happening in their surroundings in order to develop shared interpretations that can guide their actions. Thus, contrary to the emphasis in radical constructivism, our focus is not on the meaning-making activity of individuals, but on the collective generation of meaning as shaped by conventions of language and other social processes.

Table 4.1: "Best practice" approach vs. learning software organization.

Feature	**"Best practice" Approach**	**Learning Software Organization**
System perspective	Technological	Socio-technical
Human perspective	Man as an extension of the machine – an expendable spare part	Man as complementary to the machine – a resource to be developed
Skill	Maximum task breakdown, simple narrow skill – "automate"	Optimum task grouping, multiple broad skill – "informate"
Power	External control (supervisors, specialist staffs, procedures)	Internal control (self-regulating subsystem)
Configuration	Hierarchical organization	Flat organization
Leadership style	Autocratic	Participatory
Social system	Competition, gamesmanship	Collaboration, collegiality
Purpose	Organization's only	Organization's, member's and society's
Risk	Low risk-taking	Innovation

We will first focus on the concept of sensemaking itself. Then we introduce the main organizational processes of sensemaking. Finally, we examine the concept of shared assumptions, which is vital for coordinated and purposeful action in software organizations.

4.2.1 The concept of sensemaking

The concept of organizational sensemaking is to be understood literally – it means the making of sense (Weick, 1995). Thomas *et al.* (1993), for example, described sensemaking as "the reciprocal interaction of information seeking, meaning ascription, and action." (*ibid.*, p. 240). March and Olsen (1976) saw sensemaking as part of experiential learning in which "individuals and organizations make sense of their experience and modify behavior in terms of their interpretations." (*ibid.*, p. 56). Starbuck and Milliken (1988) observed that "sense-making has many distinct aspects – comprehending, understanding, explaining, attributing, extrapolating, and predicting" (*ibid.*, p. 51).

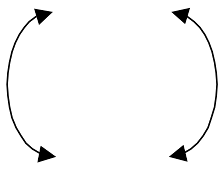
Based on his comprehensive review and discussion of the research literature, Weick (1995) identified seven properties of sensemaking understood as an organizational process, which suggests what sensemaking is, how it works, and where it can fail. First, *sensemaking is grounded in identity construction*. Thus, what the situation means is determined by the identity that the individual adopts in dealing with it. Second, and perhaps the most distinguishing characteristic, is that *sensemaking is retrospective*: "people can know what they are doing only after they have done it." (*ibid.*, p. 24). The main problem here is to select a plausible meaning from several alternative meanings in order to make sense of past events. Third, *sensemaking is enactive* since people in organizations often produce part of the environment they face, furthering the argument that sensemaking is what keeps action and cognition together. Fourth, *sensemaking is social*, since most sensemaking occurs in social groups of more than one individual. Fifth, *sensemaking is ongoing* – it never stops, but is continuous in the flow of organizational activities and projects. Sixth, *sensemaking is focused on and by extracted cues*, which provide points of reference or starting nodes from which ideas may be linked and connected into networks of meaning. Experience packages and their relationships with the context, which they are extracted from, and the context that they are interpreted into are examples of such cues. Finally, *sensemaking is driven by plausibility rather than accuracy*. Thus, organizational members behave pragmatically, as when a project deadline makes software developers trade off functionality for time.

An essential insight from the concept of sensemaking is that it can be seen as a bridge between theory and practice in SPI – much like Isenberg (1984) emphasized that managerial thinking and action are not separate or sequential activities, or like Schön (1983) emphasized the concept of "reflection-in-action". Consequently, we consider the concept of sensemaking as critically important for successful SPI.

4.2.2 Organizational processes of sensemaking

Weick (1995) described the ways in which organizational members link their thoughts and interpretations together to make collective action possible in terms of belief-driven and action-driven processes for organizational sensemaking (see *Table 4.2*).

Table 4.2: Organizational sensemaking processes (Choo, 1998, p. 78).

Belief-Driven Processes	**Arguing**: Creating meaning by connecting the contradictory.
	**Expecting**: Creating meaning by connecting the similar.
	Committing: Creating meaning to justify actions high in choice, visibility, and irrevocability.
Action-Driven Processes	**Manipulating**: Creating meaning to explain actions taken to make things happen.

> Sensemaking can begin with beliefs and take the form of arguing and expecting. Or sensemaking can begin with actions and take the form of committing or manipulating. In all four cases, people make do with whatever beliefs or actions they start with. Sensemaking is an effort to tie beliefs and actions more closely together as when arguments lead to consensus on action, clarified expectations pave the way for confirming actions, committed actions uncover acceptable justifications for their occurrence, or bold actions simplify the world and make it clearer what is going on and what it means. In each of these cases, sensemaking involves taking whatever is clearer, whether it be a belief or an action, and linking it with that which is less clear. These are fundamental operations of sensemaking. Two elements, a belief and an action, are related. The activities of relating are the sensemaking process. The outcome of such a process is a unit of meaning, two connected elements. And the connected elements are beliefs and actions tied together by socially acceptable implications (Weick, 1995, p. 135).

Belief-driven processes are those in which people construct meaning by relating past moments of socialization to present moments of experience, or cues. When such cues seem to match each other, as well as the existing frames of reference, the sensemaking process is likely to be based on "expecting". On the other hand, when cues and beliefs create a mismatch, the sensemaking process is more likely to be based on "arguing".

Action-driven processes are those in which people construct meaning around their actions. Two kinds of action can drive sensemaking: behavioral "commitment" and "manipulation". In both cases, sensemaking starts with action. The primary difference is that the commitment process is focused on single action, whereas manipulation is focused on multiple simultaneous actions.

Both beliefs and actions are potential reference points for sensemaking in software organizations. Because they are interrelated, sensemaking can start at any point in the figure in *Table 4.2*. Thus, the generation of new understandings and new actions, in whatever order they evolve, are important enablers for the contextual change necessary for successful SPI.

4.2.3 *Shared assumptions*

Sensemaking in organizations creates a context of shared beliefs and assumptions, which enables coordinated action to take place. Indeed, the consensual sharing of beliefs and behavior among members of a group is regarded as the essence of organizational "culture". Although the term *culture* first and foremost have been brought into widespread use by anthropologists (Wagner, 1981), it is also frequently used within the context of organizational learning. Schein (1992), for example, defined organizational culture as the set of shared, taken-for-granted implicit assumptions that a group holds and that determines how it perceives, thinks about, and reacts to its various environments. Furthermore, he asserted that culture manifests itself at three levels: the level of *deep tacit assumptions* that are the essence of the culture, the level of *espoused values* that often reflect what a group wishes ideally to be and the way it wants to present itself publicly, and the *day-to-day behavior* that represents a complex compromise among the espoused values, the deeper assumptions, and the immediate requirements of the situation (Schein, 1996c).

Within an organizational framework of shared beliefs and assumptions, software developers can continuously make sense of and adapt to the external environment, and continuously develop and maintain internal relationships among themselves. Peters and Waterman (1982), for example, emphasized that successful organizations build cohesive cultures around common sets of norms, values, and ideas that create an appropriate focus for doing business. However, many software organizations have short turn-around times for software developers and fragmented cultures, with multiple local realities in which shared assumptions often are based on organizational members' similar organizational experience or similar educational background.

Martin (1992) proposed three interpretive perspectives to explain organizational culture:

- The **integration** perspective, which resembles Schein's (1992) conceptualization of culture as organization wide consensus, is defined by organizational members experiencing a high level of consensus, consistency, and clarity.
- The **differentiation** perspective assumes that organizations consist of a number of subcultures based on differences in power, areas of interest, and work or professional practice. Its defining feature is that consensus only exists locally within subcultures and, thus, that collective action based on consensus is most likely to happen within local subcultures.
- The **fragmentation** perspective sees organizations as "webs of individuals" who are loosely and sporadically connected as "new issues come into focus, different people and tasks become salient, and new information becomes available." (Martin, 1992, pp. 150-151).

Additionally, in many organizations, there are typically three subcultures that have grown out of their respective occupational communities (Schein, 1996a, 1996c):

- The **operator culture** is based on human interaction, and typically involves the line managers and workers who make and deliver products and services to fulfill the organization's basic mission. In most software organization, these operations are congruent with the work of software developers, project leaders, and first line software managers.
- A key theme in the **engineering culture** is the design of the technology underlying the work of the organization. Typically, the engineers have a preference for designing humans out of the system rather than into them. This drives them toward automation and routinized solutions that often ignore the social realities of the workplace (Kunda, 1992; Thomas, 1994). As seen from the vantage point of SPI, quality managers and members of SEPGs would be representatives of the engineering culture.

- The **executive culture** is “the set of tacit assumptions that CEOs and their immediate subordinates share worldwide.” (Schein, 1996c, p. 15), and the essence of this role is financial accountability to the owners of the company. The executive culture and the engineering culture both tend to see people as impersonal resources that generate problems rather than solutions:

> One consequence is that when the operator culture attempts to improve effectiveness by building learning capacity, which requires time and resources, the executives disallow the proposed activities on the grounds that the financial returns cannot be demonstrated or that too many exceptions are involved that would undermine the control system. Executives thus unconsciously collude with the engineers in wanting to minimize the human factor. In effect, all of the research findings about the importance of teamwork, collaboration, commitment, and involvement fall on deaf ears, because in the executive culture, those are not the important variables to consider (Schein, 1996a, p. 238).

Often, it is this lack of alignment among the three cultures that causes the failures of organizational learning and SPI. Against this background, we can more easily understand the preference of formal routines within the SPI community as espoused by quality managers or members of SEPGs (see Conradi and Dybå, 2001 and *Section 8.3*). Likewise, managers will rather put emphasis on rules, procedures, and instructions than on dialog, discussion, and employee participation. Therefore, shared beliefs and assumptions are important for software organizations’ ability to relate and evaluate their actions and results.

Based on the preceding discussion, we argue that all aspects of software organizations and the relationships with their environments – whether structures, technologies, or routines – embody social constructions and meanings that are crucial for understanding and improving the software organizations’ processes. Ultimately, it is the organizations’ shared systems of meaning that govern their actions and interpretations and, therefore, their ability to learn.

As we will see in *Section 4.4.1*, software organizations must also be able to continuously assess the validity of their basic beliefs and assumptions, balancing the need for consensus and stability with diversity and experimentation. Therefore, as well as changing structures, technologies, and routines, successful SPI also depends on changes in the basic assumptions and values that are to guide action – first order change in formal systems are just not enough. Therefore, we claim that *sensemaking and organizational culture is a foundation for the learning software organization that can either hinder or enable successful SPI.*

4.3 Knowledge Creation

Within the context of SPI we are primarily concerned with organizational knowledge that enables or results in improved products, services, processes, and actions. We are particularly interested in the processes that create and diffuse knowledge since knowledge creation is a social as well as an individual process. In this section, we first examine more closely the concept of organizational knowledge itself. Then we take a closer look at individual knowledge creation processes before we examine the important features of organizational knowledge creation. Finally, we examine various enabling conditions that are important for effective knowledge creation and SPI.

4.3.1 Organizational knowledge

Knowledge is the most decisive factor in software development. Therefore, SPI strategies depend on learning, i.e. the creation of new knowledge. However, to understand and make sense of the processes of such learning we must also understand the concept of knowledge (see Dierkes *et al*., 2001, for an overview).

> To understand learning fully, we must understand the nature and forms of human knowledge and the processes whereby this knowledge is created and recreated (Kolb, 1984, p. 99).

Knowledge is both similar to and different from information. Consequently, some authors use the words “information” and “knowledge” interchangeably (e.g. Huber, 1991). Shannon and Weaver (1949) defined information as a reduction in uncertainty, or as Bateson (1979) put it, “information consists of differences that make a difference.” (*ibid*., p. 5). The view of knowledge taken by many software engineers and software engineering researchers closely resembles these definitions of information (e.g. Basili *et al*., 1994a). Thus, within the software engineering (and also the artificial intelligence) community, knowledge is most often seen as being of one kind – something tangible – something that can be acquired, shared, and distributed.

However, this tendency to treat knowledge as being of one kind severely limits both the theoretical and practical potential of SPI. Theoretically, it fails to recognize important aspects of the distinction between tacit and explicit knowledge, and between individual and collective knowledge – aspects that we consider of vital importance for understanding and improving the processes of software development. Practically, the conception of knowledge as something explicit and quantifiable draws a problematic distinction between knowledge as a tangible good and the use of that good in practice, hence limiting our ability to support practitioners in solving their real world problems.

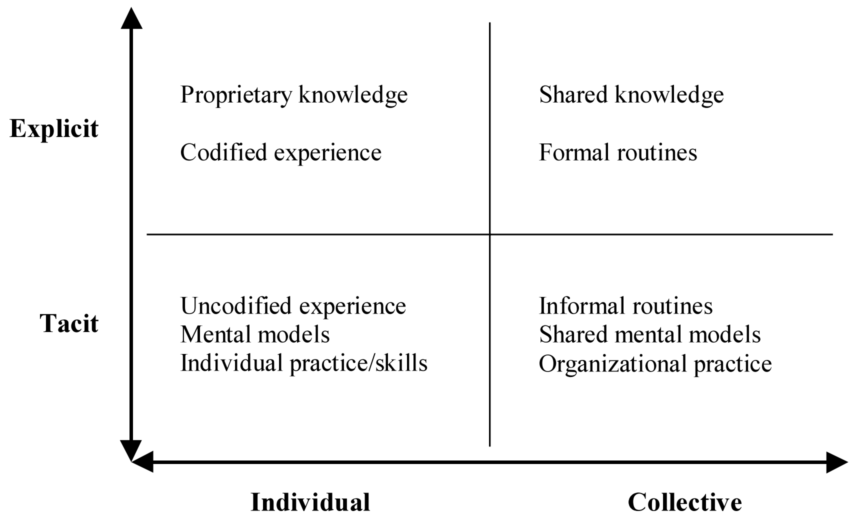


Figure 4.3: The epistemological and ontological dimensions of knowledge.

Since we are most concerned with an understanding of knowledge that has a clear link to SPI and practical work situations, we conceptualize knowledge along two dimensions (see *Figure 4.3*). The first dimension is the degree of tacit or explicit knowledge, while the second dimension relates to whether the knowledge resides in individuals, groups, or the organization as a whole. Nonaka (1994; Nonaka and Takeuchi, 1995) termed these dimensions as the "epistemological" and the "ontological" dimensions respectively.

Our understanding of the epistemological dimension is based on Polanyi's (1966) distinction between tacit and explicit knowledge. His conjecture was that "*we can know more than we can tell*" (*ibid*., p. 4, italics in original), and that knowledge that can be expressed in words and numbers only represents the tip of the iceberg of the entire body of knowledge. For example, we can recognize a person's face among thousands, but usually we cannot tell how we recognize a face that we know. Thus, explicit knowledge refers to knowledge that can be codified and expressed formally using a system of symbols, and can therefore be easily communicated or diffused (Nonaka and Takeuchi, 1995). Tacit knowledge, on the other hand, refers to implicit and uncodified knowledge, which is deeply rooted in personal experience and skills and, therefore, difficult to diffuse.

Tacit knowledge involves both cognitive and technical elements (Nonaka, 1994). The cognitive elements center on what Johnson-Laird (1983) and others (e.g. Senge, 1990) have called "mental models", which includes values, beliefs, viewpoints, and perspectives that help individuals to perceive and define their world. Conversely, the technical elements of tacit knowledge cover the concrete know-how and skills that applies to specific contexts.

Just as there is a tendency within the software engineering community to favor the explicit over the tacit, we see a similar tendency to favor the individual over the group. This is reflected, for example, by Simon (1991) who insists that all learning take place inside the heads of individuals, and also by Argyris (1999) and Argyris and Schön (1996). However, there has been a tendency toward treating groups and organizations in their own right. This is seen in the organizational literature on e.g. "communities of practice" (Brown and Duguid, 1991; Lave and Wenger, 1991; Orr, 1996; Wenger, 1998), "core competencies" (Prahalad and Hamel, 1990), "learning organization" (Senge 1990), "thinking organization" (Hunt and Buzan, 1999), "knowing organization" (Choo, 1998), "knowledge-creating company" (Nonaka and Takeuchi, 1995), and on the pursuit of "organizational intelligence" (March, 1999).

Similar to the explicit/tacit distinction, individuals and groups each do epistemic work that the other cannot (Cook and Brown, 1999). So, for example, while individual software developers have a sense of how a particular sub-module of a large program works, it is a group of developers that collectively possess the knowledge of the total system. Thus, similar to a soccer team, the body of knowledge of a software development group is possessed by the group as a whole. This leads to the contention that the organization's knowledge is not the sum of the knowledge of its individuals, but is something more and something different (Fiol and Lyles, 1985; Walsh and Ungson, 1991).

Based on the preceding discussion, we follow traditional Platonian epistemology and define knowledge as *justified true belief* (Nonaka, 1994; Nonaka and Takeuchi, 1995). Similarly, we emphasize "belief" and "justification" rather than "truth", as we regard knowledge as a social construction of reality rather than something that is true in any absolute sense. The creation of knowledge is thus not simply a matter of compiling facts; it is rather a dynamic human process that is related to the practice of individuals and groups.

4.3.2 *Individual knowledge creation*

Basically, there are two ways by which individuals acquire knowledge:

(1) Verbal transfer (by other individuals or by books, articles etc.).

(2) Direct experience (through the receipt of sensory data).

The creation of new individual knowledge through *verbal transfer* corresponds to the traditional educational way of learning which view knowledge as something that originates in the environment surrounding the learner. It is worth noting, however, that there is a world of difference between *transmitting information* (e.g. teaching) and *creating knowledge* (learning). So, while teachers, books, manuals, colleagues, or business partners can transmit information or help in arranging the conditions required for learning to take place, they cannot provide knowledge. This is also at the heart of autopoietic theory (Maturana and Varela, 1980), that knowledge cannot be imported: it can only be created.

Therefore, in order to create new valid knowledge from verbal transfer, the individual software developer must commit him or herself to actively engage in a learning process, combining the externally transmitted information with previous knowledge and intelligence. However, an important insight from several studies (e.g. Pfeffer and Sutton, 2000) is that knowledge acquired through experience is much more likely to be turned into action than knowledge acquired by reading or listening. Indeed, "The most powerful learning comes from direct experience." (Senge, 1990, p. 23).

Among the most influential theorists who have explored the central role that *experience* plays in the learning process are John Dewey, Kurt Lewin, and Jean Piaget. They inspired the work of later theorists such as Gregory Bateson, David Kolb, Chris Argyris, and Donald Schön. We have found Kolb's (1984) model of experiential learning useful in summarizing the ideas of these theorists and in explaining the process of individual learning. Kolb (1984) defined learning as "*the process whereby knowledge is created through the transformation of experience*." (*ibid*., 1984, p. 38, italics in original). What's more, he emphasized the importance of critical reflection in learning.

Kolb's (1984) theory of experiential learning summarizes the individual learning process by a cycle consisting of four stages (see *Figure 4.4*). The cycle begins with actual or "*concrete experience*" that deals with immediate human situations in a personal way. Next is the individuals' ability to reflect on and observe their experiences from many perspectives and develop these observations into collections of related ideas in a process, which Kolb (1984) called "*reflective observation*". The third stage in the learning cycle is making sense of our experiences. Kolb (1984) called this stage in which we construct models to define and explain or predict what we observe for "*abstract conceptualization*". In the final stage we seek to test our ideas in new situations through "*active experimentation*" (trying something new). The outcome of such an experiment becomes concrete experience and thus a spiraling cycle of experiential learning becomes apparent.

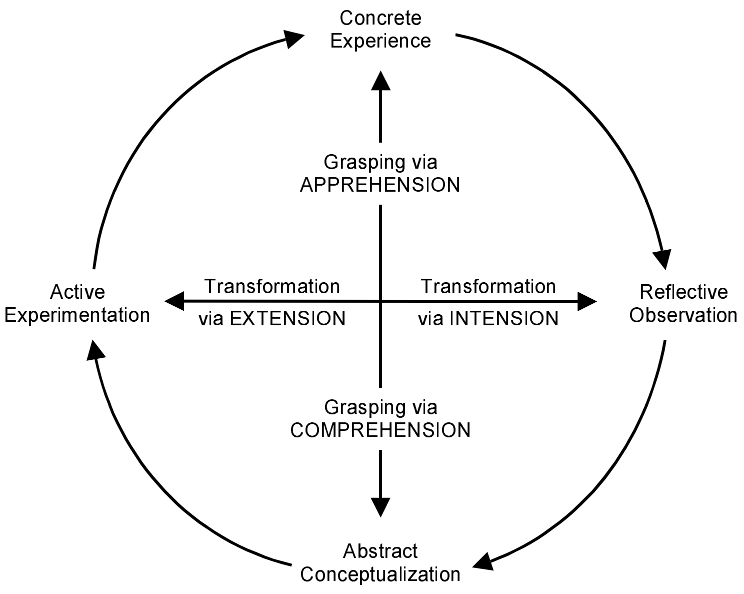

Concrete Experience
Grasping via APPREHENSION
Active Experimentation
Transformation via EXTENSION
Transformation via INTENSION
Reflective Observation
Grasping via COMPREHENSION
Abstract Conceptualization

Figure 4.4: Kolb's (1984) experiential learning cycle (p. 42).

As shown in *Figure 4.4*, there are two dimensions in Kolb's (1984) learning cycle: the prehension dimension and the transformation dimension, each representing two dialectically opposed adaptive orientations where both ends can be present at the same time. The *prehension* dimension represents the abstract/concrete dialectic and says something about how individuals are grasping or taking hold of experience in the world. The *transformation* dimension represents the active/reflective dialectic and says something about how the experiences are given meaning. Although experience is essential to learning, it is not enough; one has to do something with it to construct knowledge. The two basic dimensions of the learning process reflect this and, according to Kolb (1984), knowledge results from the combination of grasping and transforming experience.

Direct experience is also connected to processes of sensemaking, since the combination of a past moment, a connection, and a present moment of experience is what creates a meaningful definition of the present situation (Weick, 1995). Thus, experiential learning has in common with sensemaking that it requires three things: two elements and a relation.

4.3.3 Organizational knowledge creation

While Kolb's (1984) theory of experiential learning is individual-oriented, we are primarily interested in knowledge creation theory in a group and organizational context. Although the terminology varies, the same basic steps to knowledge creation appear in virtually all studies of organizational learning. For example, Daft and Weick (1984) described the process in terms of scanning, interpretation, and learning. Wikström and Normann (1994) distinguished between the generative, productive, and representative knowledge processes. Leonard-Barton (1995) identified four main activities carried out in the course of developing new products and processes through which an organization builds its knowledge and extends or creates new capabilities: (1) shared, creative problem solving; (2) implementing and integrating new methodologies and tools; (3) experimentation and prototyping; and (4) importing knowledge from the outside. In a similar vein, Garvin (2000) made a distinction between acquiring, interpreting, and applying information in order for an organization to learn.

Today, Nonaka and Takeuchi's (1995) work on the knowledge-creating company has become widely accepted as state-of-the-art in organizational knowledge creation. Based on their analysis of how Japanese companies create the dynamics of innovation, they proposed a five-phase model of the organizational knowledge creation process (see *Figure 4.5*).

Nonaka and Takeuchi's (1995) knowledge creating process starts with the *sharing of tacit knowledge*. This is a phase where individuals typically interact with each other through face-to-face dialogue in self-organizing teams in order to achieve a common goal and develop shared mental models. In the second phase, *creating concepts*, the team members collectively reflect on their shared tacit knowledge, and articulate it through further dialogue to verbalize the model into explicit concepts. In the third phase, *justifying concepts*, the newly created concepts are evaluated at the organizational level to determine whether they are in line with the organization's intention and, thus, whether they are truly worthy of pursuit. In the fourth phase, *building an archetype*, the justified concepts are converted into product prototypes or model processes. Finally, in the fifth phase, *cross-leveling knowledge*, the concepts that have been created, justified, and modeled are used to activate new cycles of knowledge creation across other groups and organizations.

Thus, "A knowledge-creating company does not operate in a closed system but in an open system in which knowledge is constantly exchanged with the outside environment." (*ibid.*, pp. 84-85).

The organizational knowledge creation cycle is based on four modes of *knowledge conversion*: from tacit to tacit (socialization), tacit to explicit (externalization), explicit to explicit (combination), and explicit to tacit (internalization). Together, these four modes of knowledge conversion feed off each other in a continuous spiral of knowledge creation.

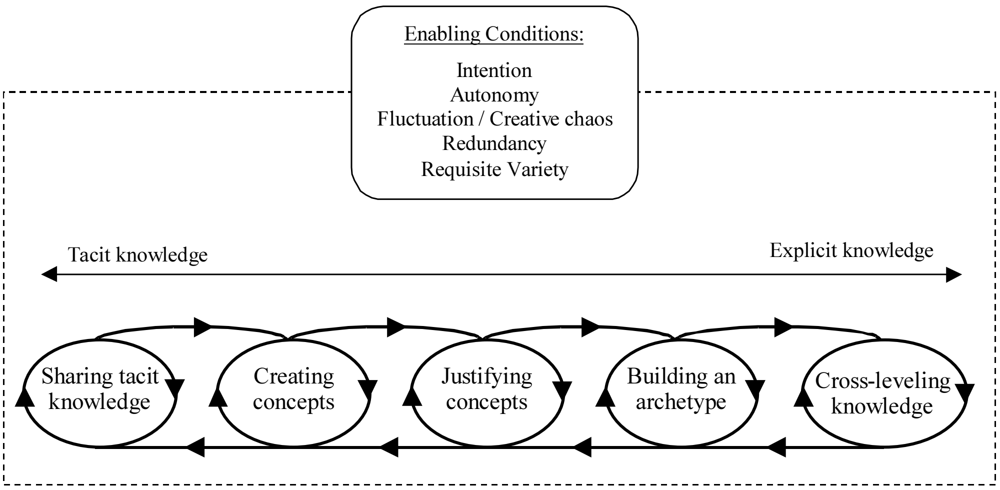


Figure 4.5: Organizational knowledge creating process (Nonaka and Takeuchi, 1995, p. 84).

However, contrary to Nonaka (1994) and Nonaka and Takeuchi (1995), whose main assumption is that knowledge is created through "conversion" between tacit and explicit knowledge, we concur with Cook and Brown (1999) and argue that one form of knowledge cannot be "converted" into the other. "Tacit knowledge cannot be turned into explicit, nor can explicit knowledge be turned into tacit." (*ibid.*, p. 385). Each form of knowledge can, nevertheless, often be used to *aid* in acquiring the other.

Thus, when a software developer tries to articulate his or her experiences of using some particular method or software engineering practice, the person in question is not performing an operation on the tacit knowledge that turns it into explicit knowledge. Rather, the tacit knowledge that comes from these experiences is used to *generate* new explicit knowledge. Therefore, the software developer still possesses the tacit knowledge, and can continue to use it in practice.

4.3.4 *Enabling conditions*

There are numerous obstacles to effective knowledge creation and successful SPI – it doesn't just happen. Therefore, *effective SPI depends on an enabling context*, similar to the Japanese concept of "*ba*" (Nonaka and Konno, 1998). Such a context can be physical (e.g. an office, or a dispersed business place), virtual (such as e-mail, teleconferencing, or web chat rooms), or mental (e.g. shared experiences, ideas, or ideals), thus providing the necessary conditions for the interaction between tacit and explicit, and individual and collective knowledge.

A concern for enabling conditions is also seen in Senge's (1990) account of the learning organization. He proposed five necessary disciplines for creating a learning organization:

- **Personal mastery** is the discipline of developing one's own proficiency and clarifying the connections between personal and organizational learning in a mutual commitment between individuals and organization.
- **Mental models** are conceptual structures in the mind that drive cognitive processes of understanding. The discipline of mental models, therefore, aims at training people to appreciate that such models do indeed occupy their minds and shape their actions.
- **Shared vision** refers to shared operating values, a common sense of purpose, or a basic level of mutuality, thus, providing a focus and energy for learning.
- **Team learning** through dialogue and discussion is considered as a vital enabler for organizational learning, since teams are considered as the fundamental learning unit in modern organizations.
- **Systems thinking** is *the fifth discipline* that provides substance to the other four disciplines. "It is the discipline that integrates the disciplines, fusing them into a coherent body of theory and practice." (*ibid.*, p. 12).

Central to his idea of the fifth discipline is that we can only understand an organizational system by contemplating the whole, and not by studying the individual parts of the system in isolation. Likewise, the enabling conditions represented by the five disciplines must develop together – as an ensemble.

As an integral part of their description of the knowledge-creating company, Nonaka and Takeuchi (1995) discussed five enabling conditions required at the organizational level to promote the knowledge spiral (see *Figure 4.5*):

- **Organizational intention**, which is an organization's aspiration to its goals. It provides "the most important criterion for judging the truthfulness of a given piece of knowledge" (*ibid*., p. 74).
- **Autonomy**, which increases the chance of introducing unexpected opportunities. This is made possible by utilizing the "minimum critical specification" principle (Trist, 1981) in self-organizing teams.
- **Fluctuation and creative chaos**, which stimulate the interaction between the organization and its environment. However, the benefits of "creative chaos" can only be realized through reflection upon organizational actions; without such reflection, fluctuation tends to lead to "destructive" chaos (Nonaka and Takeuchi, 1995).
- **Redundancy**, which refers to "intentional overlapping of information about business activities, management responsibilities, and the company as a whole." (*ibid*., p. 80), thus promoting the sharing of tacit knowledge.
- **Requisite variety**, which helps advance the knowledge spiral by matching the organization's internal diversity with the variety and complexity of the environment in order to deal with challenges posed by the environment (Ashby, 1956).

Building on Nonaka and Takeuhi's (1995) work on organizational knowledge creation, von Krogh *et al*. (2000) suggested five knowledge enablers aimed at enhancing the knowledge creation potential of a company. The first enabler, *instill a knowledge vision*, legitimizes knowledge-creation initiatives throughout the company. The second enabler, *manage conversations*, is closely connected with relationships and care in organizations, and thus strongly affects all five knowledge creation steps. The third enabler, *mobilize knowledge activists*, emphasizes the people who trigger and coordinate knowledge-creation processes. The fourth enabler, *create the right context*, is closely tied to a company's structure, since the way teams and groups are formed and interact within an organization determine the extent to which knowledge is valued. Finally, *globalize local knowledge*, emphasizes dissemination across organizational levels. *Table 4.3* summarizes Senge's (1990) five disciplines, Nonaka and Takeuchi's (1995) enabling conditions, and von Krogh *et al*.'s (2000) knowledge enablers.

Furthermore, Garvin (1993) emphasized five building blocks that learning organizations are skilled at: *systematic problem solving*, *experimentation*, *learning from past experience*, *learning from others*, and *transferring knowledge*. Recently, he has extended his analysis of the learning organization, emphasizing an "learning in action" perspective (Garvin, 2000). The latter perspective describes both the basic steps in the learning process, and three modes of learning – intelligence gathering, experience, and experimentation.

Also, based on the findings in an empirical study, Nevis *et al*. (1995) described an organization's learning system as a two-part model. One part is the *learning orientations* – the values and attitudes that determine where learning will take place and the nature of what is learned. The second part is the *facilitating factors* – the structures and processes that affect how easy or hard it is for learning to occur and the amount of effective learning that takes place. Ten facilitating factors that expedite learning were identified in their study: (1) scanning imperative, (2) performance gap, (3) concern for measurement, (4) experimental mind-set, (5) climate of openness, (6) continuous education, (7) operational variety, (8) multiple advocates, (9) involved leadership, and (10) systems perspective.

Based on the preceding discussion, we argue along the lines of von Krogh *et al.* (2000) that *knowledge creation and SPI cannot be managed, only enabled.* Therefore, software managers should focus on developing an enabling context for knowledge creation rather than trying to control the knowledge creation process. With reference to our definition of knowledge as *justified true belief* and its dynamic and human elements, knowledge depends on the situation and people involved rather than on absolute truth and hard facts. Thus, in contrast to information or data, *knowledge depends on its context and should be treated correspondingly*.

4.4 Purposeful Action

Purposeful actions are at the very heart of any SPI effort – without action, no improvement. Therefore, new interpretations and new knowledge are of limited value unless we can make use of it to construct improved courses of action. There are, however, different ways of moving toward such purposeful actions. Suchman (1987), for example, contrasted the European and Trukese approaches to open sea navigation to emphasize these differences in her discussion of technology design:

> The European navigator begins with a plan – a course – which he has charted according to certain universal principles, and he carries out his voyage by relating his every move to that plan. His effort throughout his voyage is directed to remaining 'on course.' If unexpected events occur, he must first alter the plan, then respond accordingly. The Trukese navigator begins with an objective and responds to conditions as they arise in an *ad hoc* fashion. He utilizes information provided by the wind, the waves, the tide and the current, the fauna, the stars, the clouds, the sound of the water on the side of the boat, and steers accordingly. His effort is directed to doing whatever is necessary to reach the objective. If asked, he can point to his objective at any moment, but he cannot describe his course (Berreman, 1966, p. 347, cited in Suchman, 1987, p. vii).

Table 4.3: Enabling conditions identified in organizational learning and knowledge creation.

Senge's five disciplines[1]	**Nonaka and Takeuchi's enabling conditions**[2]	**von Krogh *et al.*'s knowledge enablers**[3]
• Systems Thinking	• Intention	• Instill a Knowledge Vision
• Personal Mastery	• Autonomy	• Manage Conversations
• Mental Models	• Fluctuation and Creative Chaos	• Mobilize Knowledge Activists
• Shared Vision	• Redundancy	• Create the Right Context
• Team Learning	• Requisite Variety	• Globalize Local Knowledge

Sources: [1]Senge (1990), [2]Nonaka and Takeuchi (1995), [3]von Krogh *et al.* (2000).

This distinction between a linear and a dynamic approach to navigation could also describe a major shift from the current (old) paradigm to a new paradigm for SPI in software-intensive organizations. In the following subsections, we illustrate these modes of thinking by a description of the general theory of planned change that underlies current models of SPI (e.g. QIP and IDEAL[1]) together with the theories behind newer ways of thinking regarding change and the relationships between knowledge and action (e.g. Dybå, 2000b). However, first, we look at how these ways of thinking are reflected in contemporary theories of action.

4.4.1 Theories of action

An organization's theory of action can be defined in terms of a particular situation, *S*, a particular consequence, intended in that situation, *C*, and the organization's action strategy, *A*, for obtaining consequence *C* in situation *S* (Argyris and Schön, 1996). The general form of a theory of action is, thus: *If you intend to produce consequence C in situation S, then do A*. A path diagram of the general form of a theory of action is shown in *Figure 4.6*. Two further elements are part of this general schema: first is the values attributed to *C* that make it a desirable outcome, and second, are the underlying assumptions, or worldview, that justifies that action *A* will produce consequence *C* in situation *S*. A theory of action specialized to SPI was shown in the path diagram in *Figure 2.2* in *Section 2.1*.

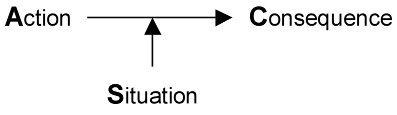


Figure 4.6: The general form of a theory of action.

There are two different forms of theories of actions: "espoused theory" and "theory-in-use" (*ibid.*). *Espoused theory* refers to the theory of action, which is advanced to explain or justify a given pattern of activity, while *theory-in-use* refers to the theory of action, which is implicit in the performance of that pattern of activity. Organizational theories-in-use may be tacit rather than explicit, and often they do not coincide with the organization's espoused theories. In Sørumgård's (1997) terms, this corresponds to a lack of process conformance. For example, a software organization's formal documents, such as organization charts, policy statements, or process definitions frequently contain espoused theories of action that are incongruent with the organization's actual patterns of activity or executed processes (see *Figure 4.7*). It is important to point out, however, that such lack of conformance need not be an obstacle for SPI. On the contrary, when such gaps between espoused theories and theories-in-use are recognized they create important opportunities for learning – opportunities that can ultimately be used to change the espoused theories, the theories-in-use, or both.

A software organization's theories-in-use largely account for its identity and culture over time. However, because of the tacitness of these theories, software developers are often unaware of the discrepancy between their espoused theories and their theories-in-use. The

[1] See *Chapter 2* for a detailed treatment of these, and other, SPI models.

reasons for this is partly that the organization's members are unable to express their tacit knowledge, and partly because a lack of conformance with the organization's espoused theories could be perceived as threatening or embarrassing. For these reasons, we emphasize that *to qualify as successful, SPI actions must include evidence of change in organizational theories-in-use* as well as in the explicitly stored organizational memory bases.

Argyris and Schön (1996) distinguished between three types of organizational learning:

(1) Inquiry that leads to improvement in the performance of organizational tasks.

(2) Inquiry through which an organization explores and restructures the values and criteria through which it defines what it means by improved performance.

(3) Inquiry through which an organization enhances its capability for learning of types (1) or (2).

The first type of learning is *single-loop learning*, which is "instrumental learning that changes strategies of action or assumptions underlying strategies in ways that leave the values of a theory of action unchanged." (Argyris and Schön, 1996, p. 20). The second type of learning is *double-loop learning*, which is "learning that results in a change in the values of theory-in-use, as well as in its strategies and assumptions." (*ibid*., p. 21). The double loop refers to the two feedback loops that connect the consequences of action with underlying values, strategies, and assumptions (see *Figure 4.8*).

Referring to *Figure 4.8*, learning occurs when a match between intentions and reality is achieved for the first time i.e. the match is genuinely new to the software organization. Learning also occurs when organizational members detect and correct mismatches or errors, thus experiencing a surprise. Accordingly, single-loop learning occurs when the mismatch is corrected by changing the actions, while double-loop learning occurs when the underlying strategies and assumptions are changed which, in turn, leads to changes in action.

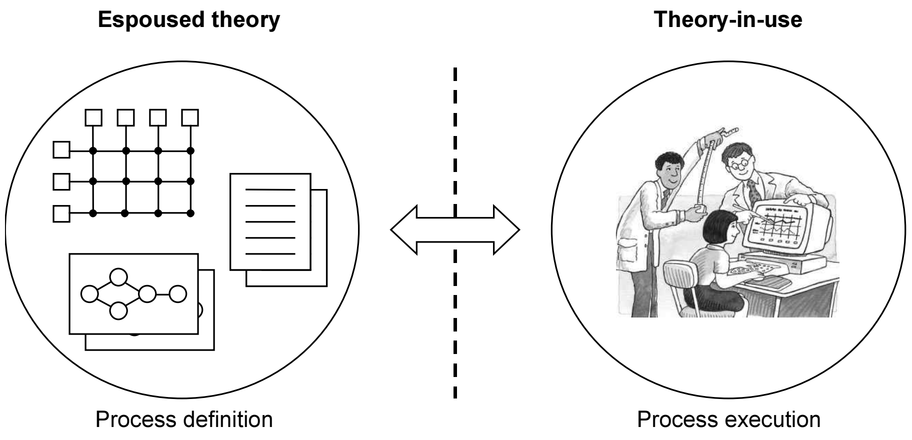


Figure 4.7: Theories of action within the context of SPI.

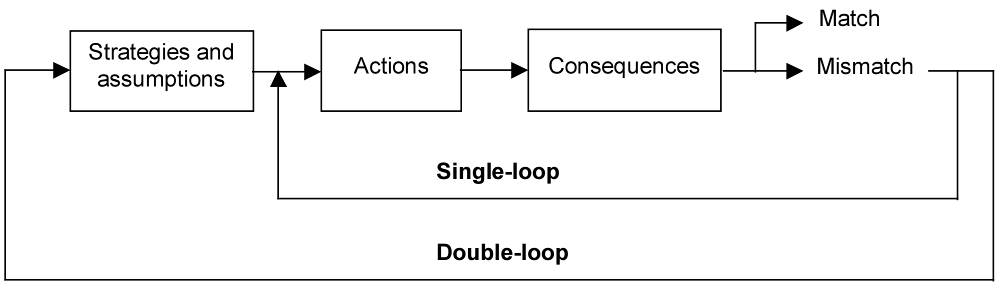


Figure 4.8: Single-loop and double-loop learning (Argyris, 1982).

The third type of learning is what Bateson (1972) called *deuterolearning*, or “learning how to learn”:

> A critically important kind of organizational double-loop learning, therefore, is the second-order learning through which the members of an organization may discover and modify the learning system that conditions prevailing patterns of organizational inquiry (Argyris and Schön, 1996, p. 29).

We should, therefore, consider efforts to develop capacities for individual and organizational learning as a key priority in designing and managing software organizations that can deal with the challenges of a constantly changing and increasingly unpredictable environment. Consequently, *the learning software organization should focus on developing skills and mind-sets that embrace environmental change*, rather than seeking an unachievable stability.

4.4.2 Models of planned change

Traditional models of planned change have their roots in Lewin’s (1951) three-stage model of “unfreezing”, “moving”, and “refreezing”. According to this model, the organization prepares for change, implements the change, and then strives to regain stability as soon as possible. *Unfreezing* usually involves reducing the forces that maintains the organization’s behavior at its present level. The process of unfreezing is often accomplished by introducing information that shows discrepancies between desired behaviors and currently exhibited behaviors in order to motivate organizational members to engage in change activities. *Moving* involves developing new behaviors, values, and attitudes through changes in organizational structures and processes. Thus, moving the organization toward a new state of behavior. *Refreezing* stabilizes the organization at the new state of equilibrium through supporting mechanisms for the new practices, such as organizational norms, policies, and structures.

Because of its broadness, several enhancements have been proposed to elaborate on Lewin’s (1951) three-stage change model (e.g. Lippitt *et al.*, 1958; Schein, 1987, 1996b). Despite its broadness, Lewin’s (1951) model of planned change has, nevertheless, been used to explain how information technologies can be implemented more effectively (Benjamin and Levinson, 1993) and how the change processes involved in SPI can be managed (Humphrey, 1989, 1998).

Greenwood and Levin (1998) countered the short-term intervention philosophy underlying Lewin’s (1951) model, arguing instead for a continuous and participative learning process consisting of two analytically distinct, but closely interrelated phases. The first phase involves the clarification of an initial research question, whereas the second phase involves the initiation and continuation of a social change and meaning construction process (see *Figure 3.3* in *Section 3.2.4*). Although their approach is different from Lewin's (1951), it still has its roots in his work. In fact, contemporary action research is still founded on Lewin's important slogans: "Nothing is as practical as a good theory" and "The best way to understand something is to try to change it."

The Burke-Litwin model of organizational performance and change (Burke, 1994) is an example of one of the more sophisticated models of planned change. The model covers both "transactional" and "transformational" change. The premise of the model is that interventions directed toward structure, management practices, and systems (policies and procedures) result in transactional change, or change in organizational *climate*. On the other hand, interventions directed toward mission and strategy, leadership, and organization culture, result in transformational change or fundamental change in the organization's *culture*.

A common feature of all models of planned change and, thus, the current paradigm of SPI is that change is viewed as a rational process that proceeds in a linear fashion from diagnosis to planning to implementation. As with Lewin's (1951) unfreeze-move-refreeze model, the beginning and ending point is stability – which, for some people and some organizations, is a luxury. For others, however, stability spells disaster – even the fastest of hares, if standing still, can be overtaken by a tortoise on the move.

4.4.3 Improvisational change models

Traditional ways of developing software and improving the software process relies on theories of planned change, which assume that the environment is predictable and that the challenges of SPI are confined to issues regarding the implementation of rational "best practice" approaches, mainly in large organizations. However, the overwhelming majority of companies developing software are small (Fayad *et al*., 2000), and for most small software organizations the environment is constantly changing and often unpredictable. Indeed, their environment is more like the croquet game in Alice in Wonderland – a game where everything is constantly changing around the player (Carroll, 1982). Furthermore, "However planned, purposeful actions are inevitably situated actions." (Suchman, 1987, p. viii).

Software organizations facing environmental turbulence have several choices (Moorman and Miner, 1998). They can ignore the external demands or turbulence that suggests the need to change plans and continue with previously planned actions. They can attempt to speed up their planning and execution cycles so that they remain distinct but happen more quickly (Eisenhardt and Tabrizi, 1995). Or they can move toward an improvisational approach that merges planning and execution processes. We consider such an *improvisational* approach as fundamental for understanding the relationship between action and learning in software organizations. For that reason, we contend that *software organizations need to become more improvisational in order to survive in an increasingly turbulent and complex environment* (see Dybå, 2000b).

Improvisation deals with the unforeseen, and involves continual experimentation with new possibilities in order to create innovative and improved solutions outside current plans and routines. However, organizational improvisation does not emerge from thin air. Instead, it involves and partly depends on the exploitation of prior routines and knowledge.

"Improvisation involves reworking precomposed material and designs in relation to unanticipated ideas conceived, shaped, and transformed under the special conditions of performance, thereby adding unique features to every creation." (Berliner, 1994, p. 241). Such "smart" or "competent" improvisation was also central to Ciborra's (1999) theory of information systems. Hence, improvisation involves a mix of exploiting previously learned lessons and exploring the possibilities and contingencies of the current setting. This mix, however, points to a core paradox of improvisation, which is also at the heart of SPI: *Too much reliance on previously learned patterns tends to limit the explorative behavior necessary for improvisation; while too much risk-taking leads to fruitless experimentation and repeated failures* (Dybå, 2000b).

As an example of an improvisational change model used in the introduction of new software technologies, we give a short overview of the model devised by Orlikowski and Hofman (1997). Their model rests on two major assumptions that differentiates it from the traditional models of change. First, change is perceived as an ongoing process rather than an event with an end point after which the organization returns to a new state of stability. Second, all the technological and organizational changes made during the ongoing process cannot, by definition, be anticipated ahead of time.

Given these assumptions, Orlikowski and Hofman's (1997) improvisational change model recognizes three different types of change, which build on each other iteratively over time (see *Figure 4.9*): anticipated, emergent, and opportunity-based. *Anticipated changes* are changes that are planned ahead of time and occur as intended, like, for example, organizational improvement programs (Dybå, 2000c) or process improvement experiments (Dybå, 2000d). *Emergent changes*, on the other hand, are changes that arise spontaneously from local innovations and that were not anticipated or intended. Finally, *opportunity-based changes* are changes that were not anticipated ahead of time but are introduced purposefully and intentionally during the change process in response to an unexpected opportunity, event, or breakdown. The feedback sessions in GQM is an example of a context from which opportunity-based changes originate (Dybå, 2000e). Thus, both anticipated and opportunity-based changes involve deliberate action, while emergent changes arise spontaneously and usually tacitly from people's practice with the technology over time.

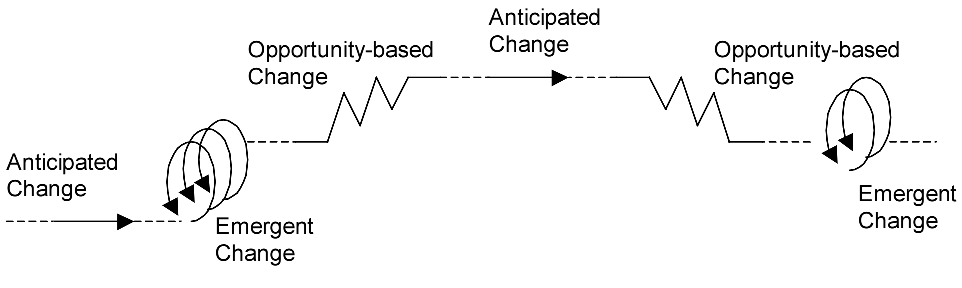


Figure 4.9: An improvisational model of change (Orlikowski and Hofman, 1997, p. 13).

The notion of improvisation occurs in several contexts such as music, theatre, teaching, therapy, and sports. Recently, the *jazz metaphor* has been introduced as a source of understanding organizational processes of creativity and innovation, and of interpreting complex and changing environments (Barrett, 1998).

> While members of a jazz band, unlike members of a symphony orchestra, do not decide in advance exactly what notes each is going to play, they do decide ahead of time what musical composition will form the basis of their performance. Once the performance begins, each player is free to explore and innovate, departing from the original composition. Yet the performance works because all members are playing within the same rhythmic structure and have a shared understanding of the rules of this musical genre. … Using our earlier terminology, the jazz musicians are engaging in anticipated, opportunity-based, and emergent action during the course of their performance to create an effective, creative response to local conditions (Orlikowski and Hofman, 1997, p. 13).

Similar to the performance of a jazz band, an improvisational model for SPI is not a predefined improvement plan charted by management ahead of time. Instead, it recognizes that improvements stem from an iterative series of many unpredictable changes that evolve from practical experience with new methods and tools. Therefore, we want to emphasize that improvisation at the local level is not only a legitimate alternative to models of planned change, but also necessary for effective knowledge creation and SPI. This does not mean that we abandon plans and planning altogether. On the contrary, we consider plans as valuable resources for purposeful action, or as Suchman (1987) formulated her view of plans as "formulations of antecedent conditions and consequences of action" (*ibid.*, p. 3). What we *are* suggesting, however, is that a plan should be acknowledged as a guide rather than a blueprint and that deviations from the plan, rather than being seen as a symptom of failure, are to be expected and actively managed. Planning in the presence of environmental turbulence should be regarded as an ongoing endeavor, reflecting the changing, unfolding environments with which software organizations interact (Mintzberg, 1994) and not as a one-shot exercise. *Creating an organizational context of enabling conditions is therefore of critical importance for an improvisational approach to succeed.*

4.4.4 Organizational knowing

So far, we have maintained the predominant view of knowledge as something that is *possessed* despite our expanded understanding of it suggested by the tacit/explicit and individual/group dimensions (see *Section 4.3.1*). For example, when we say that "Alice has knowledge of software development", knowledge refers to something that Alice possesses (e.g. as concepts, methods, and procedures). However, while knowledge itself is seen as static, it is customary to think of it as something we *use* in action but it is not normally understood to *be* action itself. In a perspective where we consider knowledge as a social construction we should rather use the term "knowing", which also is closely connected to improvisational models of change.

Hence, in accordance with Choo (1998) and Cook and Brown (1999), we take a pragmatic[2] standpoint and use the term "knowing" to refer to the epistemological dimension of action itself. That is, contrary to knowledge, "By 'knowing' we do not mean something that is *used in* action or something *necessary to* action, but rather something that is a *part of* action (both individual and group action). … *Knowing is that aspect of action (or practice) that does*

[2] See *Section 6.1* for a justification of the pragmatic approach in our research.

epistemic work." (Cook and Brown, 1999, pp. 387, 388, emphasis in original). Thus, when we talk about Peter's "knowing" in software development, we focus on what he is actually doing – e.g. how he deploys his knowledge in the programming of a specific algorithm.

> Accordingly, when it comes to questions of what we know and how we know, the Pragmatist perspective takes a primary concern not with 'knowledge,' which is seen as abstract and static, but with 'knowing,' which is understood as part of concrete, dynamic human action (*ibid*., p. 387).

Knowing, therefore, is closely related to theory-in-use, which means that if we want to understand the essentials of how to improve software development, we must pay attention to what software developers do (their "knowing") as well as to what they possess (their "knowledge"). Furthermore, we must recognize that knowledge, once possessed, is not enough to enable knowing or action. "As a tool, knowledge disciplines knowing, but does not enable it any more than possession of a hammer enables skillful use." (*ibid*., p. 388). Likewise, knowledge of the principles of SPI cannot by itself enable a software organization to engage in the activity of improving its processes and enhance its competitiveness.

Consequently, we argue that in order to succeed with SPI we must move from a conceptualization of knowledge as an object to be possessed toward a broader view of knowing as "an ongoing process of social construction and collective action" (Choo, 1998, p. 224) that is embedded in the software organizations tasks, relationships, and tools. This is similar to the properties defined by Schön's (1983) regarding "knowing-*in*-action".

Pfeffer and Sutton (2000) also focused on turning knowledge into action in their study of the "knowing-doing gap". Therefore, software organizations should embed more of their knowledge acquisition activities in the "real work" of developing software and less in formal training programs and benchmarks against so-called "best practice" models, thus putting knowledge back into the context in which it has meaning.

4.5 Chapter Summary

In this chapter, we have described our perspective on organizational learning and SPI as a collective ability expressed in and through the organizations' product- and service-oriented practices related to software development. This perspective led us to emphasize a more adaptive and active view of knowledge rather than the traditional static and passive view.

Furthermore, we recognized four foundations for the learning software organization: *social learning*, focused on SPI as a socially constructed, collaborative activity; s*ensemaking*, aimed at constructing meaning and to express basic assumptions and values; *knowledge creation*, aimed at generating new knowledge and new competencies; and finally, using the new interpretations and new knowledge to construct improved courses of *purposeful action*.

Based on the insights derived from these foundations, the next chapter describes a dynamic model of SPI, which embraces the learning processes and the key factors for success required for describing software organizations as learning systems. A major concern is to establish a conceptual framework, which structures the organizational learning process, thus enabling the diffusion of knowledge and experience within and between groups of software developers.

CHAPTER 5

A Dynamic Model of Software Process Improvement

"*Those who cannot remember the past are condemned to repeat it.*"
– George Santayana

In the previous chapter, we argued that successful software process improvement requires a commitment to learning and that organizational knowledge is constructed through the collaborative action of organizational members. Therefore, to explain why some SPI initiatives are more successful than others are, we need a model of SPI that incorporates the foundations of the learning software organization.

In this chapter, we propose a dynamic model of SPI for examining a software organization's learning capability and the processes that enable sensemaking, knowledge creation, and purposeful actions within the context of a learning software organization. The model is grounded in our personal experience and observations in the case studies, related to the extensive literature review, and focused on the three research questions posed in the introductory chapter: What are the key learning processes in successful software organizations? What are the key factors of success in software process improvement? What are the relationships between organizational context and modes of learning in software organizations?

Furthermore, the model presented in this chapter includes the *theoretical justifications*, detailed conceptualizations, and hypotheses for the key factors of success in SPI. In addition to these theoretical justifications, we conducted exploratory studies in four of the case companies to empirically justify the hypotheses (*Case D*). Finally, we arranged a review of the combined theoretical and *empirical justifications* by performing a survey among SPI experts from both academia and industry (see *Figure 6.3* in *Section 6.2*). The details of the empirical justifications and the results of the expert review are described in *Chapter 7*, together with the details of the operationalizations necessary for testing the hypotheses. The hypotheses are statistically formulated and tested in *Section 8.4*.

5.1 Introduction

Software-intensive organizations that intend to excel in the twenty-first century must learn to manage change in dynamic situations. Current models of SPI, which are founded on the old "unfreeze-move-refreeze" paradigm are, as we argued in the previous chapter, insufficient to guide change in a constantly changing and increasingly unpredictable environment. Thus, rather than seeking an unachievable stability, *the learning software organization should focus on developing routines and mind-sets that embrace environmental change*.

The model developed in this chapter supports this shift by directing attention to the needs for *communication*, *coordination*, and *collaboration* within and between software teams. The model is about how software teams acquire and use knowledge in an organizational setting in order to improve their software processes. Verbs like "knowing" or "learning" are used to emphasize the action oriented and dynamic properties of SPI and the noun "knowledge" is used to describe the static properties.

The model can be seen as following the fundamental principle of the hermeneutical circle (Klein and Myers, 1999), by attempting an initial holistic understanding of an organizational system and then using this understanding as a basis for interpreting the parts of the system. According to this model, knowledge is gained dialectically by proceeding from the whole to its parts and then back again. Each time an incongruence occurs between part and whole, a reconceptualization takes place. The frequency of such reconceptualizations decrease as the match improves between the conceptualization of the organization and that held by the organization's members.

The model can also be seen as following a pragmatist viewpoint on SPI and ESE research. According to the model, the knowledge that the software organization accumulates, its methods for accumulating it, and the criteria by which these methods are considered valid are all based on the organization's prior experience for dealing with what Dewey (1929) called "problematic situations." The methods used and the criteria for appraising them as valid are matters of social convention. As situations, which the organization considers problematic, change, so may its methods for dealing with them and the criteria for judging them as valid.

This viewpoint suggests that a software organization want to know more about the world and engage in SPI activities when it is uncertain about achieving desired outcomes. This occurs because the organization does not understand the situations in which it wants to take action, or because it is uncertain about what action to take to achieve them. This uncertainty about situations or what actions to take in them is what makes them problematic and is the point from which SPI begins. This is very different from the model-based approaches described in *Chapter 2* in which SPI is seen as starting with the implementation of "best practices" according to a predetermined scheme, independent of the organization's experience of problematic situations.

A critical element in our model, therefore, is the integration of SPI activities with the "real work" of software development. This way, we consider software teams and their projects as the baseline for improvement (see Dybå, 2000d) and the first line manager and the software developers who develop the systems as the foremost responsible for keeping the organization's processes on the leading edge of technology.

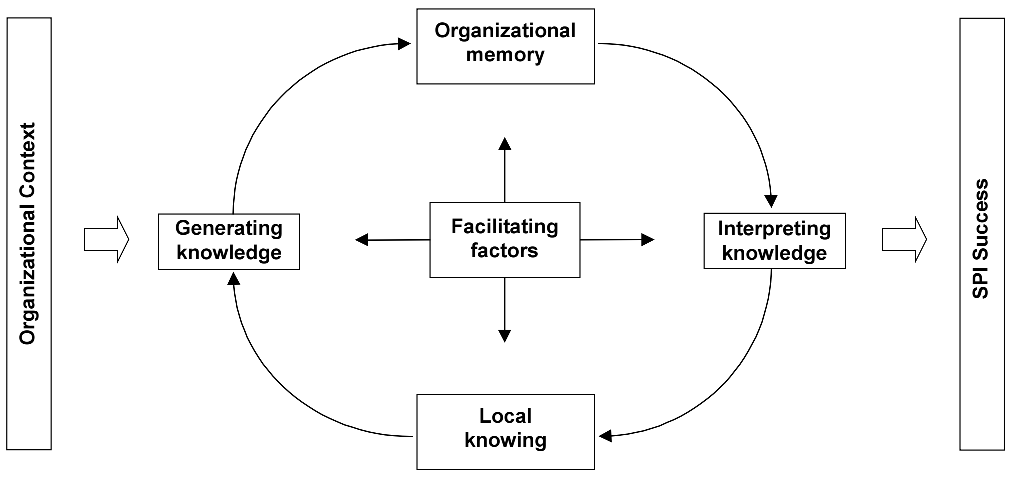


Figure 5.1: A dynamic model of software process improvement.

Figure 5.1 presents an overview of the dynamic model of SPI, which explicitly addresses the research questions asked in the introductory chapter of the thesis. In addition to the foundations discussed in the previous chapter, the structure of the model is inspired by systems thinking (Senge, 1990) and open systems theory (Katz and Kahn, 1978) in which an organization is viewed as an input-throughput-output system.

The model contains the following four major elements:

- **Organizational context**. This is the general environment that imposes constraints and opportunities about what the organization can and cannot do with respect to organizational learning and SPI. Since we perceive the organization as an open system, the reality experienced by the various software teams contains elements from outside the organization as well as from the organization itself (see *Section 5.2*).
- **Learning cycle**. The organization's learning cycle is a dialectic process that integrates local experience and organizational concepts. All members and groups of members in the organization contribute in the social construction of the software organization's knowledge. At the same time, the organization's memory limits the range of the possible actions for its members. The process of generating new organizational knowledge and collectively interpreting the knowledge is, thus, a simultaneous and dynamic one (see *Section 5.3*).
- **SPI success**. This is the performance or results of the organization's SPI activities. This is the dependent variable that is used to measure that gains have in fact been made with respect to organizational behavior and performance and not merely at the cognitive level (see *Section 5.4*).
- **Facilitating factors**. These are the conditions that facilitate or enable knowledge creation and SPI. They are the key factors for success that the software organization must put in place in order to facilitate the organization's learning cycle and improve its development process (see *Section 5.5*).

According to this model, organizational learning and SPI is defined as a dynamic interplay between two primary dialectics. The first is that between the local and organizational level. The other is that between generating and interpreting organizational knowledge. The dialectic between local knowing and organizational memory is the interplay between the knowing of the organization's members in their respective contexts and the knowledge that the organization has established over time. This interplay is a dynamic, two-way relationship between the organization and its members, which combines local transformation with the evolution of the organization. This is similar to Piaget's (1970) description of the learning process as a dialectic between *assimilating* experience into concepts and *accommodating* concepts to experience. In our model, organizational learning results from a balanced tension between these two processes. Our emphasis is thus on SPI as a dialectic process that integrates local experience and organizational concepts.

The model presented in this chapter has several advantages. First, it should be clear that organizational knowledge is not being created to mirror a reality that is independent of human action, but to deal with it. Second, starting SPI from problematic situations in software teams reduces the risk that organizational learning will be detached from action, and undertaken to build knowledge for its own sake. Third, it increases the likelihood that knowledge intended for application to practical problems will ultimately serve that purpose since knowledge gained from concrete situations is more likely to remain applicable to future concrete situations.

5.2 *Organizational Context*

Generally, the quality management literature supports the proposition that ideal quality management should not be affected by contextual variables. Juran and Godfrey (1999), for example, stated that ideal quality management is "universal" and suggested that the expectations regarding quality management should be the same regardless of the context "no matter what is the industry, function, culture, or whatever" (*ibid.*, p. 2.5). Crosby (1979, 1996), Deming (1986), and Feigenbaum (1991) also support this context-free view of quality management. However, empirical studies have indicated that organizational context, nevertheless, do influence manager's perceptions of both ideal and actual quality management, and that contextual variables are useful for explaining and predicting quality management practices (Benson *et al.*, 1991).

Like most of the quality management approaches, a context free view of process improvement is at the heart of the "best practice" paradigm to SPI. This paradigm is rooted in Taylor's (1911) principles of task management, which he believed was equally valid for all classes of work, "from the most elementary to the most intricate" (*ibid.*, p. 40), and for all human activities, declaring that his principles were applicable "from our simplest individual acts to the work of our great corporations." (*ibid.*, p. 7). Taylor's view was by no means unique. Henry Ford, for example, believed in the general applicability of the principles of mass manufacturing (Morgan, 1997), and William Henry Leffingwell was "obsessed with the notion of bringing rational discipline to the office in much the same way that Taylor and his men were attempting to transform the shop floor." (Zuboff, 1988, p. 117). Subsequently, the functionally specialized procedural work model was adapted for administrative and clerical work in form of the bureaucratic organization and, furthermore, to software development and SPI in form of "best practice" models like CMM, ISO/IEC 15504, Trillium, and Bootstrap.

In contrast to the "best practice" or model-based approach to SPI, the analytic approach described in *Chapter 2*, is more concerned with the contingent characteristics of individual organizations. For example, the importance of context is made explicit in the different steps of QIP and also in the various templates and guidelines for the use of GQM. In addition to the measurement object, template parameters for setting goals in GQM include purpose, quality focus, viewpoint, and environment (Basili and Weiss, 1984). Furthermore, the definition of contextual parameters is a formalized part of the GQM abstraction sheet, where one part of the questions and metrics is devoted to "quality focus", while the other part is devoted to questions and metrics regarding "environmental" or contextual variables (Gresse *et al*., 1995). Finally, and most importantly, is the evaluation of the impact of the contextual variables and how they may affect the organization's improvement practices (see *Table 8.1* for an example).

Despite important differences, both the model-based and analytical approach to SPI seem to be most concerned with solving the needs of *large organizations* operating in highly *stable environments* with long-term contracts (e.g. the U.S. Department of Defense and the NASA/GSFC). This is further confirmed by famous cases of successful SPI such as Alcatel (Debou *et al*., 1999), Hewlett-Packard (Grady, 1997), Hughes (Humphrey *et al*., 1991), Motorola (Daskalantonakis, 1992), Philips (Rooijmans *et al*., 1996), Raytheon (Dion, 1993), and Siemens (Mehner, 1999), which are veritable giants compared to small and medium-sized enterprises (SMEs).

Representing the model-based approach, the authors of the CMM made it clear that it was developed for use with large, government-funded type of projects (Paulk *et al*., 1995). It can be tailored and the documentation to do so is available from the SEI (Ginsberg and Quinn, 1995). However, the process of tailoring is itself so complex and resource demanding that it is of questionable value to most SMEs (Ould, 1996).

What's more, Humphrey (1989) put the case for statistical control of the software process, arguing that "If the process is not under statistical control, sustained progress is not possible until it is" (*ibid*., p. 3). Moreover, the SEI has advocated the use of SPC techniques, specifically control charts, in recent technical reports and proposed changes for the CMM to explicitly support the use of rigorous statistical techniques (Florac and Carleton, 1999). The main prerequisites for the application of SPC, however, are stable processes and lots of data. Although expert knowledge is an important part of using the GQM (see *Section 8.1*), emphasis in the analytical approach is still on large organizations generating large amounts of data.

These concepts are not of much help for most SMEs facing increasingly changing environments where the periods of stabilization are constantly shortened. As the rate of change increases, the complexity of the problems facing SMEs also increases. The more complex these problems are, the more time it takes to solve them. The more the rate of change increases, the more the problems that face SMEs change and the shorter is the life of the solutions they find to them. Therefore, by the time they find solutions to many of their problems, the problems have changed so that the solutions to them are no longer relevant or effective. In other words, many of the solutions are to problems that no longer exist in the form in which they were solved.

Consequently, most SMEs face two challenges: (1) an ever-changing environment and (2) few projects running at any given point in time. As a result of this, they have few data, which they can analyze and use to build up an experience base. In addition, collected data will soon be outdated and left irrelevant or – in the best case – uncertain. Taken together, this implies that SMEs cannot base their improvement actions on collecting long time series or amass large amounts of data needed for a tradition statistical improvement approach.

However, the special needs of SMEs have been recognized by both the EFQM and by the European Community through ESSI. As a result, the EFQM launched the European Quality Award for SMEs in 1997, which is for both whole companies and independent business units inside larger companies. Within the ESSI program, several hundred SMEs have performed Process Improvement Experiments (PIEs) to demonstrate and disseminate the benefits of SPI through user experimentation.

Thus, two contextual variables are included in the model to capture the most influential sources of variation and to provide greater quasi-experimental control: *environmental turbulence* and *organizational size*.

5.2.1 *Environmental turbulence*

The software organization's environment refers to various characteristics outside the control of the organization that is important to its performance. These characteristics include the nature of the market, political climate, economic conditions, and the kind of technologies on which the organization depends.

The environment of a particular software organization may range from stable to dynamic – from predictable to unpredictable. In a stable environment the software organization can predict its future conditions and rely on standardization for coordination (Mintzberg, 1989). Certainly, a stable environment may change over time, but the variations are still predictable. But when the conditions become dynamic – when the market is unstable, the need for product change is frequent, and turnover is high – such change is highly unpredictable and the software organization cannot rely on standardization. Instead, it must remain flexible through the use of direct supervision or mutual adjustment for coordination, and so it must use a more *organic* structure. Therefore, the effectiveness of a software organization's structure will depend on the environment of the organization.

5.2.2 *Organizational size*

The organizational literature suggests that large organizations are less likely to change in response to environmental changes than small organizations. Tushman and Romanelli (1985), for example, argued that increased size leads to increased complexity, increased convergence, and thus, increased inertia. Likewise, Mintzberg (1989) postulated that the larger an organization, the more formalized its behavior. So, while small organizations can remain organic, large organizations develop bureaucracies with job specialization and sharp divisions of labor, emphasizing stability, order, and control. As a consequence, they often have great difficulties in adapting to changing circumstances because they are designed to achieve predetermined goals – they are not designed for innovation.

From a learning perspective, however, inertia develops as a result of the organization's performance history (Lant and Mezias, 1992). Large organizations tend to be successful since an organization will grow larger with repeated success. However, since success reduces the probability of change in a target-oriented organization (Cyert and March, 1963, 1992), *large software organizations will be less likely to change when the environment changes*.

5.3 Learning Cycle

As we have already argued, organizational learning and SPI is defined as a dynamic interplay between two primary dialectics. The first is that between the local and organizational level. The other is that between generating and interpreting organizational knowledge. In this section, we make a detailed description of each of these four elements of the learning cycle.

5.3.1 Local knowing

The primary context, within which meaning is constructed, new knowledge created, and improved courses of action taken, is the shared practice within local software development teams. Software developers do not work in isolation – they work together to develop products that they could not develop by working as individuals. This focus on teams and their collaborative processes are important because no one developer embodies the breadth and depth of knowledge necessary to comprehend large and complex software systems, and also because codified or explicit organizational knowledge is seldom sufficient to solve a particular problem. Thus, just as a single soccer player cannot play a game of soccer by him or herself, only a group of software developers, working as a team, can develop software of a certain size and complexity.

The software teams' way of grasping the world and forming local realities is by apprehension – in the present movement of "here-and-now" (Kolb, 1984). They are concerned with concrete situations as experienced in all their complexity during software development. They act in a specific context in which reality is constantly being created and recreated. Local reality is therefore not an explicated and static model of causal relationships for software development. Rather, it shows up in the local actions taken by the developers in the team and can, as we argued in the preceding chapter, better be characterized as "knowing".

Thus, by *local knowing* we refer to the knowledge-in-action associated with participating in the collective practice of software development in a specific context. It is important to stress this, since a software organization's primary concern is the actual *practice* of developing software, and not merely the creation of knowledge on how to do it. Local knowing is, therefore, about how the software organization works, or its theories-in-use, as seen from the local teams or work groups in the organization. Participating in software teams is consequently not only a matter of developing software, but also to change the organization's knowledge about software development and to generate software process improvements.

The context in which software developers interact contributes to the learning process in several ways. First, each software team or work group operates in a particular setting with a particular mix of people, tools, and techniques to define and solve a particular software development problem. Also, the way in which software developers use prior experience and available tools and techniques varies with the particular, concrete circumstances. That is, software developers will approach a certain problem depending on the actual setting because each setting tends to evoke certain kinds of "appropriate" modes of thought and action (Tyre and von Hippel, 1997). Moreover, software developers often take advantage of the setting itself to help them define a problem or to discover solutions.

Also, software developers incorporate codified organizational routines into local, informal practices, freely adapting the routines as they work on solving actual problems in their particular circumstances. Local knowing draws on both the organizational members' individual understandings of the situation and their ability to use the relevant parts of organizational memory that is available in a given context. Therefore, the context in which software development takes place partly determines what the organization's members can do, what they know, and what they can learn. Moreover, since different local settings provide different opportunities for learning, any SPI activity will also be a *situated* process.

Therefore, all software development and SPI activities have an ad hoc adroitness akin to *improvisation* because they mix together the contingency of the present situation with lessons learned from prior experience (Dybå, 2000b). Ryle (1979) described this mixture as "paying heed", to be thinking at what one is up against here and now by adjusting oneself to the present situation while at the same time applying the lessons already learned. In other words, local knowing is affected by the current setting as well as by the organization's memory of its past experience.

Such an improvisational theory of local knowing has its roots in the pragmatists' notion that knowledge is not absolute, but rather can only be defined in relation to a specific situation or context (Dewey, 1938). Questions about what is "true" are answered in relation to what works in a given setting. Consequently, local knowing is *pragmatic* and produces actions that are oriented toward established goals, directed at doing whatever is necessary to reach the objective.

Thus, *organizational learning and SPI occurs through people interacting in context* – or, more precisely, in multiple contexts. This situated and pragmatic characteristic of organizational learning has important implications for how problem framing, problem solving, and SPI take place in software organizations. Most importantly, this perspective suggests that traditional, decontextualized theories of SPI cannot completely account for learning in software organizations. Rather, since learning is an interactive, social process, contextual factors will affect both how and what organizational members learn.

There are several social groups within a software organization that share knowledge and that may be identified as having a distinct local reality. Examples of such groups are formal project teams and informal groups of software developers and managers. A group's local reality can be seen as a way of acting in relationship to the rest of the organization. However, *shared practice by its very nature creates boundaries* (Wenger, 1998).

There are two basic conditions for establishing connections across such boundaries and making communications between the groups effective. First, each group must respect the expertise of the other, and must acknowledge the relevance of that expertise to their own problems. Second, each group must have sufficient knowledge and understanding of the other groups' problems to be able to communicate effectively about them. However, experience shows that these conditions are unlikely to be satisfied unless a sufficient number of members of each group have had actual experience with the activities and responsibilities of the other groups (Simon, 1991).

The most important groups influencing SPI in software organizations are:

- *Software developers and first line software managers* in general have a common work experience from software development. Although their responsibilities differ and their work is done in different departments, they have a similar background and work in the same chaotic reality. Their understanding of software development and SPI is closely related to specific events and their knowledge is highly contextualized. More often than not, they express their knowing in stories through face-to-face contact with their peers.
- *Quality managers* seldom have a software background, but they are, nevertheless, expected to provide software developers and software managers with standards and routines that can improve the quality of their organizations' products and services. Their understanding of software development and SPI is typically influenced by the manufacturing professions, and they continuously try to decontextualize the lessons that software developers have learned. Furthermore, they espouse a firm belief that SPI can be achieved by the institutionalization of a quality management system based on common standards and explicit routines.
- *Executive managers* have an overall and general understanding of the organization and an overall accountability for organizational performance. Often, they seem to have little contact with the way work is actually performed in the organization's diverse teams. Their knowledge of software development is decontextualized and they have little ability to directly influence actual work practices. They often focus on economy and, like quality managers, they are concerned about order, stability, and repeatability in software development. Incidentally, this is also the core assumptions of most textbooks in software engineering and of most research on software development methods and tools.

Rather than order and stability, the reality experienced by most software development teams is more akin to chaos due to the *complexity of work* combined with a dynamic and increasingly *unpredictable environment*. So, while executive managers often favor direct supervision and quality managers want standardization as the organization's main coordinating mechanisms, mutual adjustment is the only operational mechanism that we have observed in actual use within and between creative software teams. This is also in agreement with Mintzberg's (1983, 1989) postulate that mutual adjustment is the only coordination mechanism that can handle really complex work with a lot of problem solving.

Mutual adjustment depends, in large, on physical, face-to-face contact, which is the richest communication channel we have and by far the most effective form of transferring and exchanging knowledge and experience in local teams. Also, physical, face-to-face experience and interaction are the keys to create and diffuse tacit knowledge. Therefore, people working together with frequent, easy contact will more easily exchange knowledge and experience with each other than people that are separated by time and space. This has important implications for SPI, since local software development teams can utilize the flexibility of face-to-face communication and shed bureaucracy.

However, the communication capacity rapidly becomes saturated as the group grows, and without compromises, it is impossible to extend mutual adjustment in its pure form to organizations larger than the small group. In large organizations, real mutual adjustment can only take place within development teams or groups small enough to allow all-to-all communication.

Nevertheless, with the support of proper technology, considerable extension of the coordination of work, within and between teams, by mutual adjustment is possible if the adjustment is mediated by indirect communication through a repository of externalized organizational memory. Such *implicit coordination* (Groth, 1999) of software developers working from a common experience base greatly reduces the need for extra communication and direct supervising efforts in the organizational learning process. Contrary to efforts of providing better tools for handling the increased communication, such as groupware solutions or efforts at standardizing the work process, the attack point in our model, which is based on experience bases and computerized organizational memory, is to *reduce the volume of communication needed for coordination.*

Contrary to Nonaka and Takeuchi's (1995) understanding of organizational knowledge creation as a process of "cross-leveling" across hierarchical levels in an organization, we consider various forms of organizational memory to be the most important mechanism for implicit coordination and learning *between* teams in software organizations (see Conradi and Dybå, 2001 and *Section 8.3*).

In the next section, we describe the process of generating new explicit knowledge based on local knowing so that lessons learned can be incorporated in organizational memory and shared outside the team.

5.3.2 Generating explicit knowledge

Generating new explicit knowledge is a *collective* process where a group of software developers attempts to externalize their local knowing. This means, for example, that a software team must take time to express its shared practices in a form that can meaningfully be understood and exploited by other organizational members. This process involves the articulation of tacit knowledge into such explicit knowledge as concepts, models, and routines through the use of words, metaphors, analogies, narratives, or visuals. The result of this process is new organizational knowledge and an extended range of explicit organizational memory.

All externalization processes depend on language, and writing is a fundamental act of generating new explicit knowledge. Language is the means by which we articulate shared experience and generate new organizational knowledge. It provides the means for incorporating newly articulated knowledge into organizational memory. Another important part of the externalization process is the face-to-face interaction that yields stories and anecdotes of experience, much like the photocopier repair technicians in Orr's (1990, 1996) study based their externalization of experience on story telling.

In practice, dialogue (Bohm and Peat, 2000) and collective reflection (Schön, 1983), or reflective observation to use Kolb's (1984) terminology, triggers the articulation of explicit knowledge. This process of generating new explicit knowledge brings some of what the software team apprehends into what the team comprehends.

Dialogue is an important way of collectively grasping experience through comprehension such that the software team is able to articulate and build models of their experience and thereby communicating it to others. To the extent that such experience models are accurately constructed from the team's local knowing, they allow others to predict and recreate that knowing.

Collective reflection and dialogue facilitate a greater coverage of past experience, since individual developers can prompt each other to help remember the past. In this sense, multiple and even conflicting individual experiences enable a more comprehensive recollection of past events. Such diversity in local knowing between software teams should not be seen as a problem, but rather as a valuable source for organizational learning and SPI. Thus, "it is the differences, not the agreements, that are the possibilities for learning and change." (Gjersvik, 1993, p. 197).

One of the most effective ways of externalizing local knowledge in software organizations is through the use of models, tools, and techniques. An example of such a process is how consulting firms develop models of the software development lifecycle by trying to capture and codify the tacit knowledge of their consultants. Typically, the consultants participate in project teams who remain and work at the client site for months or even years to develop custom-build applications. The consultants document their projects in project files, which, collectively, serve as the company's experience base. As the company and the number of projects grow, attempts are usually made to systematize this varied and highly context specific experience and to extract the common success and failure factors in order to generalize rules of thumb for good software development practice. Over time, these informal guidelines evolve into standard lifecycle models, which define the sequence of tasks in each stage of the models as well as the conceptual framework within which information systems should be developed for specific clients. The lifecycle models, thus, grows out of the daily activities of consultants working on client projects. Through such formalization processes, therefore, we are able to externalize much of the tacit knowledge of experienced consultants into explicit models of the software development process.

When constructing models or systems, however, only parts of the local reality will be externalized since "The program is forever limited to working within the world determined by the programmer's explicit articulation of possible objects, properties, and relations among them." (Winograd and Flores, 1986, p. 97). Such modeling creates a blindness that is limited to what can be expressed in the terms that the organization has adopted. Although this is an unavoidable property of models and technology, the software organization should, nevertheless, be aware of the limitations that are imposed.

Most often the blindness that is created by the use of computer systems is a blindness for other possible local realities than the rationalistic tradition. Furthermore, it seems that groups holding local realities, which are close to the rationalistic tradition more easily get their externalizations through in their respective organizations. Incidentally, executive managers and quality managers are the ones in software organizations that most easily can be associated with this tradition. Their local realities are more easily externalized not just because it is easier to externalize an abstract and decontextualized view of the organization rather than the tacit assumptions of the developers' context-based knowing, but also because they can take advantage of the organization's asymmetrical distribution of power. Since managers have a greater influence on decision-making and allocation of resources than other organizational members, it is easier for them to externalize themselves and make their worldview dominate what is considered important for the organization's memory.

We have used several knowledge-creation techniques as part of this study to externalize, evaluate, and organize new knowledge. Among the most widely used was the *GQM approach* described in *Chapter 2*, the *KJ Method* (Scupin, 1997), and *Mind Maps* (Buzan and Buzan, 2000). Common to these techniques is that they help a group of developers to create ideas and articulate their knowledge through two phases. During the *divergent thinking phase*, the participants articulate key words, phrases, goals, questions, or metrics which they think are relevant for the dialogue. In GQM, these concepts are documented in GQM abstraction sheets, while the KJ method uses less structured Post-it Notes, and Mind Maps uses a picture of words.

During the *convergent thinking phase*, groups using GQM combine their abstraction sheets into one sheet per goal, and jointly try to resolve any conflicts and inconsistencies. With the KJ method, the participants organize their Post-it Notes into logical groups, which are then related into a diagram of concepts and ideas as the conclusion. In a similar way, Mind Maps are used to organize concepts by placing each idea next to what it is related to.

This dialectic of divergent and convergent inquiry facilitates the surfacing of hidden assumptions. The collaborative nature of these processes and the utilization of figurative language for concept creation are what, in our experience, make these techniques so powerful tools for collectively externalizing the tacit knowledge of a group of software developers and, thus, generating new organizational knowledge.

Articulating tacit knowledge and creating new, explicit concepts is not enough. For new knowledge to be useful for others outside the team, it must also be packaged. Knowledge gained locally should be consolidated and globalized in the form of *experience packages* and stored in an Organizational Memory Information System (OMIS), or Experience Base, so it is available for future projects. In principle, most kinds of experience can be externalized, packaged, and made available in the organization's experience base, for example (Basili *et al*., 1994a):

- Product packages (e.g. programs, architectures, designs).
- Process packages (e.g. process models, methods).
- Relationship packages (e.g. cost and defect models, resource models).
- Tool packages (e.g. static analyzers, regression testers).
- Management packages (e.g. management handbooks, decision support models).
- Data packages (e.g. project databases, quality records).

Still, each organization must decide for itself what knowledge needs to be packaged based on its business values and needs. Furthermore, since face-to-face interactions need to be high when transferring new concepts to a different location, each experience package should be indexed with local areas of expertise and references to groups or individuals who can help the receiving unit. Moreover, the organization should decide on how its experience packages should be stored in organizational memory.

However useful techniques a software organization might use for the articulation of explicit knowledge and experience packaging, the local knowing can never be fully represented in organizational memory. Contextual information is inevitably lost in this process and what is stored in organizational memory will be a decontextualized subset of local knowledge. Therefore, proper consideration of how memory objects will be decontextualized and then recontextualized in future use is necessary. We must, in other words, be able to consider the present through the lens of future activity (Ackerman and Halverson, 2000).

In the next section, we describe the process of incorporating experience packages into organizational memory together with examples of typically observed memory categories.

5.3.3 Organizational memory

Organizational memory is a generic concept used to describe an organization's capability for adoption, representation, and sharing of new beliefs, knowledge, or patterns for action. It is essential for organizational learning and SPI to occur by embedding organizational members' discoveries, inventions, and evaluations. Sometimes this may require official action and issuing revised regulations or operating guidelines. However, since each local group within an organization has its own culture, it also requires informal acceptance by enough opinion leaders and rank and file members for it to be disseminated as valid and valued knowledge.

In other words, that which is accepted in one part of an organization may or may not be passed on to other units or parts of the organization – one unit's knowing could be another unit's rubbish or heresy. Thus, lessons-learned cannot easily be transferred from one setting to another. Also, higher levels of the power structure can destroy the learning of lower levels as a matter of policy, or even as a matter of neglect or indifference – except sometimes in the case of a strong counter-culture arising out of long conflict and shared grievance. Memories are, thus, cooperatively created and used throughout the organization and, in turn, influence the learning and subsequent actions that are taken by the local groups in the organization.

Each time a software organization restructures itself, the contents of its memory are affected. Since much of the organization's memory is stored in human heads, and only a little of it is put down on paper or held in computer memories, *turnover of personnel is a great threat to long-term organizational memory*. When experts leave, the costs to the organization are even greater because it takes years of education, training, and experience to become one (Simon, 1991). Loss of such knowledge can undermine the competence and competitiveness of the organization, and can also have a serious impact on cultural norms and values.

Thus, a better understanding of organizational memory and its role in organizational learning can help software organizations in framing and solving problems related to the utilization of organizational knowledge and the improvement of their software development processes. This is also at the heart of the Experience Factory, which was discussed in *Chapter 2*. However, we should be careful not to assume that the availability of organizational memory necessarily leads to organizations that are effective; it can also lead to lower levels of effectiveness and inflexibility (Weick, 1979).

Based on Walsh and Ungson's (1991) definition, we focus on organizational memory as *the means by which a software organization's knowledge from the past is brought to bear on present activities*. This definition makes no assumptions regarding the impact of organizational memory on organizational effectiveness since this depends on the ways in which the memory is brought to use. For example, when organizational knowledge is consistent with the goals of the organization, organizational memory can be said to contribute to organizational effectiveness. At the other extreme, organizational memory can be seen as a structure that objectivates a fixed response to standard problems that constrains and threatens the viability of organizations operating in turbulent environments.

Table 5.1: Memory categories and examples of typical elements.

Memory Category	Typical Elements
Worldview	Culture, beliefs, assumptions, values, norms, strategies, power relations, symbols, habits, expectations
Structure	Task structure, roles, behavior formalization, coordinating mechanisms, unit grouping, workplace ecology
Plans and models	Life cycle models, assessment models, project plans, milestone plans, quality plans, improvement plans, measurement plans, action plans
Systems	Information systems, tools and techniques, quality control systems, training systems, social systems
Routines	Rules, standard operating procedures, development processes
Lessons learned	Experience reports, articles, memos, newsletters, stories, feedback sessions, peer reviews, post mortem reviews

Therefore, the members of the software organization must themselves determine what to do with the knowledge they acquire in order to meet the incompatible demands of change and stability. Organizational memory can be viewed as a structure that both enables action within the software organization by providing a framework for common orientation and, at the same time, limits the range of action by constraining the possible ways of developing software. Thus, just as organizational memory provides stability, it can also serve to block change.

To be useful for the software organization as a whole, newly created concepts have to be communicated and explained to others who have not shared the concrete experience. This means that the new concepts must be turned into justified true beliefs at the organizational level. This then, makes *justification* an essential process since the organization must decide whether new concepts and beliefs are worthy of further attention and investment (von Krogh and Grand, 2000). There is an inherent dialectic here that the justification process tries to balance. On the one hand, newly generated knowledge has to be related to existing organizational knowledge in order to be acceptable and understandable. On the other hand, new knowledge challenges the organization's existing understanding of the world through its novelty, provoking complex processes of argumentation and justification, to be decided in favor of the existing or the newly emerging views.

Justification processes are therefore important for the software organization's memory since they decide whether new knowledge is *rejected* as irrelevant or uninteresting, *returned* to the local team for further elaboration, or *appropriated* as justified true belief and therefore integrated into organizational memory.

Apart from the knowledge held by individual members of the organization and their collaborations, we have observed several categories of organizational memory. These memory categories and examples of typical elements are shown in *Table 5.1* and described next:

- ***Worldview***. The software organization's worldview is the culture or shared assumptions that govern what kind of experience is believed to be useful for dealing with the future. Therefore, the organizational and technological choices made by an organization in order to improve its software development processes are critical expressions of its worldview. Typically, the worldview of a software organization is formally expressed in business strategies and policy statements or implicitly assumed by the content and structure of organizational routines.

- ***Structure***. The main characteristics of a software organization's structure are the ways in which its software activity is divided into various tasks and how these tasks are coordinated to develop software. Examples of organizational structures are the departments and units of the organization, its systems, its models and plans, its meetings, and its coffee breaks. Also, roles provide an effective structural mechanism by functioning as a distributed repository of task knowledge and, thus, as a link between individual and organizational memories.

  Workplace ecology is also an important part of organizational memory that should be considered in any change process, which helps shape and reinforce behavior prescriptions within the organization (Walsh and Ungson, 1991).

- ***Plans and models***. According to the rationalistic tradition, plans and models are prerequisite to action and prescribe, at every level of detail, how the action should be performed to accomplish some preconceived end. However, the rationalistic approach is not suited to situations where the mind and the motivation of the worker matter more than his or her ability to do simple, routine tasks. As in software development and SPI in which the work requires the knowledge and experience of the developers, it also requires the developers' motivation, involvement, and commitment to succeed. Removing control over the development from the developers has the eventual effect of destroying this motivation.

  Contrary to the rationalistic perceptions of a linear relationship between planning and action, SPI actions often occur without much advance planning. Indeed, the industrial improvement plans developed in SPIQ did not in any strong sense determine the software organizations' courses of action. Consequently, we should not limit the concept of organizational memory to formal knowledge such as software manuals, employee handbooks, training material, etc., but also appreciate informal knowledge such as tacit know-how, expertise, experience, stories, etc., which are most often ignored in software organizations. Therefore, the key is not getting the right plans or models but enabling knowledge creation.

- ***Systems***. Like plans and models, systems are also part of the rationalistic heritage. Taylor (1911), for example, claimed that "In the past the man has been first; in the future the system must be first" (*ibid*., p. 7). The most notable systems in software organizations are, as we pointed out in *Chapter 1*, the *social system* of software developers cooperating with each other and the *technical system* of computers, tools, methods, techniques, and control systems. Radically improved performance in software organizations can, therefore, only be achieved through "the joint optimization of the social and technical systems." (Trist, 1981, p. 24), creating a synergy that yields more than could be achieved simply by adding the two together. Moreover, "Attempts to optimize for either the technical or social system alone will result in the suboptimization of the socio-technical whole." (*ibid*.).

  Like Huber (1991), we acknowledge the importance of *information systems* in serving organizational memory. It is important to note, however, that the software organization has a choice regarding its use of such systems. Rather than being designed for control purposes, such systems should be designed to enable experiential learning and the use of technology to enhance the skills of the developers rather than to deskill them (Corbett, 1992) – to "informate" rather than to "automate" tasks (Zuboff, 1988).

- ***Routines***. Organizational routines are a major source of organizational knowledge. They are important because a lot of organizational actions are performed and coordinated through routines (Cyert and March, 1963, 1992; March and Simon, 1958, 1993). They are also important because the organizations' accumulated experience often takes form in a set of routines that can outlast the particular organizational members who first developed them (Levitt and March 1988).

  The foremost use of routines arises in repetitive situations – used in inappropriate situations they can have a serious problematic effect. They are like a "two-edged sword" (Cohen and Bacdayan, 1994) – they allow efficient coordinated action, but also introduce the risk of highly inappropriate responses.
- ***Lessons learned***. Most software organizations do not document their lessons learned. However, large parts of these lessons are still made explicit through peer reviews, feedback sessions, postmortem analyses, and the stories told by organizational members during face-to-face interaction. This part of organizational memory is also perceived by the developers to be the most important and relevant for dealing with the complex situations of actual practice (see Conradi and Dybå, 2001 and *Section 8.3*). A key ingredient of the Learning Software Organization, therefore, is that it is able to learn from experience rather than being bound by its past.

For a software development team to be able to reuse a memory object like an experience package, it must be recontextualized and made relevant for the new situation. That is, the memory object must be reunderstood for the developers' current purpose. A proper understanding of how local knowing is first decontextualized and adopted as organizational memory and then recontextualized into new local knowing is of critical importance for the utilization of organizational memory. This problem has largely been unnoticed in contemporary debates on experience bases within SPI, which is often limited to the technical challenges of implementing a database. However, if we do not address the problems of recontextualization, the whole concept of organizational memory and experience bases will be more or less useless.

The next section describes how the organization's memory can be put back into use and become part of local knowing through a process of collective interpretation.

5.3.4 Collectively interpreting the knowledge

The collective interpretation of knowledge is the process of making organizational memory an integral part of local knowing by making sense out of the actions, systems, structures, plans, models, and routines in the organization. Through this process, the organization's memory is recontextualized and taken up into the practice of local software development teams. It is a process of "re-experiencing" (Nonaka and Takeuchi, 1995) other team's experiences.

A major confusion in much of the thinking in contemporary knowledge management and SPI is to equate easy access of information with learning. However, there is an important difference between passively receiving information and actively interpreting and making sense of it. When an individual software developer receives information, he or she relates that information to past moments of experience in order to make sense of it. And, as we noted in the previous chapter, it is the differences from what is expected, and not the agreements, which provide the possibilities for organizational learning and SPI. Therefore, we attend to that which is different from our current understandings and from our expectations in order to compare it with already extracted cues. Learning can only be said to have taken place when the individual has formed new networks of meaning and new reference points for future sensemaking processes from the information encountered.

Collective interpretation processes are still more complex. Not only must each software developer engage in an individual process of sensemaking, he or she must do so while simultaneously interacting with other developers. By engaging in *collective interpretation*, each developer is influenced by the meanings held by others and in turn influences the meanings of others. This way, each developer can better understand the experiences and reasoning the other developers are using in their interpretations and by comparison, understand each other's meanings more fully. Based on these interactions, the software developers are in a position to form a collective interpretation of the organizational knowledge that is available to them.

Therefore, collectively interpreting organizational knowledge involves *active construction of knowledge* in the form of active formulation and solution to problems with the help of explicit models, guiding routines, and feedback. This highlights an important aspect of the social construction of SPI: collective interpretation is effective not necessarily as a function of simple internalization, with modeled information being transferred across a barrier from the organization to the inside of a team, or with information being transmitted. Rather, these interpretations are effective through peripheral and active participation (Lave and Wenger, 1991), whereby the members of a team collectively transform their understandings and skills in framing and solving a problem. According to this view, it is the active construction through firsthand experience that is so crucial to learning and SPI, not some distant guidance or universal rule. It should thus be clear why a constructivist approach should be treated as a close ally to organizational learning and SPI.

Rather than being transmitted or internalized, *knowledge becomes jointly constructed in the sense that it is neither handed down ready-made from the organization, nor something a team constructs purely on its own*. Knowledge, understandings, and meanings gradually emerge through interaction and become distributed among those interacting rather than individually constructed or possessed. Furthermore, since knowledge is distributed among participants in a specific activity context it is, as we have already argued, necessarily situated as well. That is, intimately welded to the context and the activity in which and by means of which it is constructed. Importantly therefore, *participation* becomes the key concept here, as contrasted with acquisition, conceptual change serving as both the process and the goal of learning.

In the process of forming collective interpretations, it is important that we distinguish between reducing *ambiguity* and reducing *uncertainty*. Ambiguity is the lack of clarity about the technologies and processes of software development when the environment is difficult to interpret, and when cause and effect are disconnected so that the organization is unable to link its actions to their consequences. It has more to do with the confusion of multiple meanings than with the absence of sufficient quantities of information. The lack of meaning drives sensemaking, while the lack of certainty drives data collection and information gathering: "In the case of ambiguity, people engage in sensemaking because they are confused by too many interpretations, whereas in the case of uncertainty, they do so because they are ignorant of any interpretations." (Weick, 1995, p. 91). Thus, approaches to measurement-driven SPI can support the reduction of uncertainty, but they don't necessarily assist the software organization in reducing the ambiguity that is essential for organizational learning and SPI.

The process of "re-experiencing" other team's experiences involves experimenting with organizational knowledge in local contexts by "giving it a try". Based on the concepts of ambiguity and uncertainty we can distinguish between two types of such experiments that are crucial for organizational learning and SPI: hypothesis-testing experiments and exploratory experiments. *Hypothesis-testing experiments* are field experiments designed to reduce the organization's uncertainty by discriminating among alternative explanations or solutions from many possibilities. This is the usual way of conducting process improvement experiments according to the experimental approach described in *Chapter 3*. Of special concern to us here, therefore, is the conduct of exploratory experiments to reduce ambiguity.

Exploratory experiments involve learning through discovery, encouraging the flexibility and resilience needed to cope with the situation at hand. When ambiguity is high, the knowledge represented by the organization's memory provides little support. So, during this phase of the learning cycle the focus shifts from justification and exploitation of existing knowledge, to skepticism and exploration of new opportunities.

Such exploration is of utmost importance in unfamiliar and ambiguous situations. But such exploration or "learning by doing" only works when a team receives rapid and unambiguous feedback on its actions. However, in the complex reality experienced by most software teams, the consequences of their actions are neither immediate nor unambiguous. In these situations, effective learning can, nevertheless, be achieved by the use of simulated environments, what Nonaka and Konno (1998) termed "exercising *ba*", or "microworlds" to use Senge's (1990) terminology. In such microworlds, it becomes possible for software teams to learn about future and distant consequences of their actions by experimenting in environments that "compress time and space" (Senge, 1990).

Prototypes are important examples of microworlds within organizational learning and SPI that enable the collective interpretation of knowledge. Developing a prototype is an experimental activity mainly concerned with reducing the inherent uncertainty and ambiguity of specifications (Mathiassen and Stage, 1992), thus facilitating a shared understanding of the system to be developed.

There are two main approaches to exploration in SPI in which prototypes serve an important role: probing and learning, and pilot projects. In *probing and learning* the software organization constructs "quick-and-dirty" mock-ups. To be useful for the learning process, these prototypes still have to be close enough approximations of the final product or development process. Otherwise, such experimentation will be of little value since generalizations will be virtually impossible. Furthermore, the probing and learning process should be designed as an *iterative* process, since it is hardly possible to "get it right the first time" in an ambiguous environment.

Pilot projects are projects aimed at on-line experimentation in real software projects or large-scale simulations in separate demonstration projects (see Dybå, 2000d). Typically, pilot projects are the first projects to embody principles and approaches that the organization hopes to adopt later on a larger scale. They implicitly establish policy guidelines and decision rules for later projects. They often encounter severe tests of commitment from employees who wish to see whether the rules and practices have, in fact, changed. They are normally developed by strong multifunctional teams reporting directly to senior management. And, finally, they tend to have only limited impact on the rest of the organization if they are not accompanied by explicit strategies for the diffusion of experience and knowledge gained from the pilot projects (Garvin, 1993, 2000).

The context dependent inferences of prior experience and memory objects can only be carried over from one organizational situation to another through a kind of "seeing-as" (Schön, 1983). When a software team makes sense of a situation it perceives to be unique, it *sees* it *as* something already present in the repertoire represented by organizational memory. Therefore, "Seeing *this* situation as *that* one, one may also *do* in this situation *as* in that one." (*ibid*., p. 139, italics in original).

> It is our capacity to see unfamiliar situations as familiar ones, and do in the former as we have done in the latter, that enables us to bring our past experience to bear on the unique case. It is our capacity to *see-as* and *do-as* that allows us to have a feel for problems that do not fit existing rules. ...
>
> Reflection-in-action in a unique case may be generalized to other cases, not by giving rise to general principles, but by contributing to the practitioner's repertoire of exemplary themes from which, in the subsequent cases of his practice, he may compose new variations (*ibid*., p. 140).

Consequently, in order to learn and improve their software processes, software teams can sometimes figure out how to solve unique problems or make sense of puzzling phenomena by modeling the unfamiliar on the familiar. Depending on the initial proximity or distance of the two things perceived as similar, the familiar may serve as an "exemplar" or as a "generative metaphor" for the unfamiliar (Schön, 1983). In both cases, the software team arrives at a new interpretation of the phenomena before it by "reflecting-in-action" on an earlier perception of similarity.

The utility of an experience package lies in its ability to generate explanation and experimentation in a new situation. When the experience package is carried over to the new situation, its validity must be established there by a new round of experimentation through which it is very likely to be modified. The modified experience package that results from this new round of experimentation may, in turn, serve as a basis for transfer and re-creation to a new situation.

So, for organizational learning and SPI to happen, organizational members must act on the collective interpretations they have made – starting a new cycle of organizational learning. Thus, purposeful action at the local level is a means for the interpretation of organizational knowledge as well as for the generation of new knowledge and, consequently, essential for organizational learning and SPI.

5.4 SPI Success

Defining the dependent variable in any study of SPI is of utmost importance (DeLone and McLean, 1992). However, despite an increasing number of studies trying to identify those factors that contribute to SPI success, the dependent variables in most of these studies have been hard to pin down or they are without adequate theoretical and psychometric justification. For example, Stelzer *et al*. (1996) and Stelzer and Mellis' (1998) studies of success factors have unclear definitions of SPI success, and no quantitative data on the relationships between success factors and success.

In their studies of CMM experience and results, Goldenson and Herbsleb (1995) and Herbsleb and Goldenson (1996) defined SPI success in terms of organizational performance in addition to a single item on how well the assessment findings were addressed. They defined organizational performance with respect to six performance characteristics: ability to meet schedule and budget commitments, product quality, productivity, moral, and customer satisfaction. But they did not include these measures of organizational performance in their analysis of the factors that influenced success.

Like Goldenson and Herbsleb (1995) and Herbsleb and Goldenson (1996), both El Emam *et al.* (1999) and El Emam *et al.* (2001) used the same single-item dependent variable related to how successfully the findings and recommendations of an assessment had been addressed to model the likelihood of SPI success. However, as we discuss in *Chapter 7*, such single-item measures tend to be highly unreliable and, consequently, of questionable value in determining the factors for SPI success (also see Dybå, 2000a; Nunnally and Bernstein, 1994; Spector, 1992).

On the other hand, in their assessment of the effectiveness of software processes on project performance, Deephouse *et al.* (1996) defined project performance in terms of two dimensions: software quality and meeting targets, which included schedules and budgets. Furthermore, Sawyer and Guinan (1998) used three performance dimensions in their study of the effects on software development performance due to the production methods of software development and the social processes of how software developers work together. The dimensions were stakeholder-rated product quality, stakeholder-rated team performance, and self-reported (by the developers) team performance. Each of the three performance dimensions were operationalized using multi-item scales that were shown to have high factor reliabilities.

Also, Goldenson *et al.* (1999b) used two multi-item dimensions of success as the dependent variable in a survey on the determinants of success in software measurement programs. Both dimensions, use of measurement and organizational performance, were found to have satisfactory psychometric properties.

5.4.1 Organizational performance

We argue that *organizational performance is the ultimate criterion for any SPI activity*. Performance is a complex construct, however, reflecting the criteria and standards used by decision-makers to assess the functioning of a software organization. That is, performance is a value judgement on the results desired from an organization (van de Ven and Ferry, 1980). This is also reflected in the goal-oriented strategies to software measurement and SPI described in *Chapter 2*. For our purposes, we need a joint measure of performance that can be used across a multitude of organizations to compare the relationships between SPI success and the facilitating factors.

Despite espoused strategies about customer relationships, core competencies, and organizational capabilities, the assessment of organizational performance has, traditionally, focused on long term profitability and financial measures. For many software organizations, this focus on profitability has turned into a general concern about productivity and, consequently, on measuring success in such terms as the number of lines of code produced per individual per day. However, neither profitability nor productivity indicators are sufficient to predict future competitiveness in the software business. In today's technologically and customer-driven global competition, financial measures of performance often provide poor guidelines for success. This realization was also the main driver behind the development of the Balanced Scorecard (Kaplan and Norton, 1996, 2000), which highlights a more general and integrated set of measurements that link current customers, internal processes, employees, and system performance to long-term financial success.

As a fundamental part of our model, therefore, we need a dynamic concept of SPI success that represents a software organization's competitiveness. Performance, which is something an organization does (process) or achieves (outcome), is a concept that better can serve as an operational tool for improvement of competitiveness than pure productivity or profitability measures (Bredrup, 1995).

Furthermore, satisfied customers is an important asset for a software organization, it is the cornerstone of any TQM program, and it is the most important principle in the recent revision of ISO 9000:2000. Therefore, the customer perspective should be a central part in any model of a software organization's performance.

Lynch and Cross (1991) defined customer satisfaction as the difference between the customers' perceived performance and their needs and expectations:

$$\textit{Customer satisfaction} = \textit{Perceived performance} - \textit{Expectations}$$

A classic problem, however, is that both performance and expectations are subjective terms and that performance as seen from the software organization, can be viewed differently than performance as seen from the customer. Typically, the customer will focus on *external performance measures* such as price and delivery time, while the software organization focuses on *internal performance measures* such as cost and lead-time. The relationships between such external and internal performance measures are, therefore, critical for the integration of customer satisfaction in any model that purports to measure SPI success. However, improved profitability is not an automatic outcome of SPI programs to improve customer satisfaction.

All software processes are expected to deliver a quality product on schedule and on budget in order to achieve customer satisfaction and thereby to ensure long-term profitability for the software organization. Moreover, these fundamental characteristics have importance to both customers and the software organization and they are, therefore, important for the understanding and definition of SPI success. In other words, SPI should lead to "better, faster, [and] cheaper software development" (Sanders, 1998). This is also clear in Krasner's (1999) model of the challenges in software development projects, which focuses on the dynamic relationships between software processes and the three outcome factors: cost, schedule, and quality.

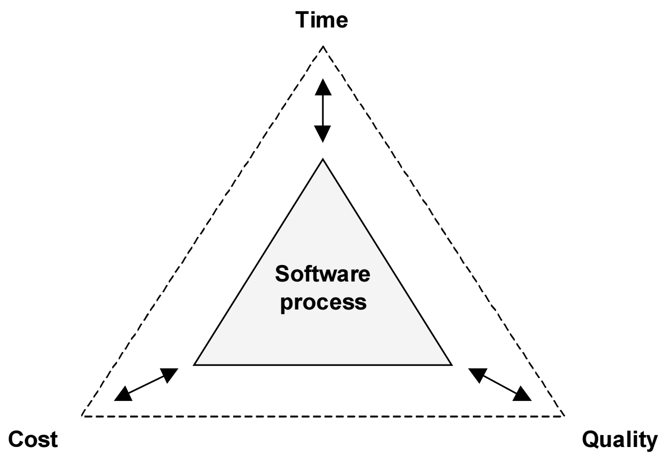


Figure 5.2: The organizational performance dimension of SPI success.

From the preceding discussion we have identified organizational performance as an important dimension in the measurement of SPI success. Furthermore, we have identified the following three elements as central constituents of organizational performance as seen from a customer satisfaction perspective (see *Figure 5.2*):

- ***Time***. Time to market has become a critical measure for software organizations in today's turbulent markets. Being able to respond rapidly and reliably to customer requests and changing market conditions is often critical for a software organization's competitive-ness. Including time-based metrics as part of the organizational performance measure, therefore, signals the importance of achieving and continually reducing lead-times for meeting targeted customers' expectations. Yet, other customers may be more concerned with the reliability of lead times than with just obtaining the shortest possible lead-time. In addition to lead-time or cycle-time reductions, therefore, measures of on-time delivery rate improvement and schedule slippage rate reductions can also be useful time-based indicators of customer satisfaction and retention.
- ***Cost***. Customers will always be concerned with the price they pay for products and services. Long-term profitability, therefore, requires that there is a healthy relationship between price and cost and, consequently, that we include process cost metrics as part of the organizational performance measure. Process cost includes the cost of the primary activities (marketing and sales, inbound logistics, operations, outbound logistics, and service) and the support activities (infrastructure, human resource management, technology development, and procurement) in the software development value chain (Boehm and Papaccio, 1988). Although the major source of software costs is the "operations" component, virtually all components are still highly labor-intensive. Therefore, effort is frequently the predominant cost driver for most software processes. Examples of potentially useful cost metrics are: ratio of actual versus planned cost of work effort, development hours saved, productivity increases, rework cost reduction, and reuse increases.

- ***Quality***. Using the Kano model (Kano *et al*., 1984) as the frame of reference, we have witnesses a tendency among large customer groups that quality is not always expressed as an explicit requirement – it is so obvious that it is often not even mentioned. Nevertheless, the customers' expectations consist of both the explicitly stated, *functional* and *non-functional*, requirements and the obvious implicit, or tacit, requirements. This is also the focus of the quality definition in ISO 8402:1994:

  Quality is the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.

  However, in certain parts of the software industry, the situation is such that excellent quality may still offer opportunities for companies to distinguish themselves from their competitors. In any case, customer perceived quality is always relevant to include as an organizational performance measure. Examples of such quality metrics are defect density reductions, and customer satisfaction increases. An important part of this picture, however, is that the software organization may not even be aware of the unsatisfied customers – they simply cease to use the organization's products or services. Interestingly, an American study revealed that 96 percent of unhappy customers never tell the company (Kotler, 1988).

To summarize, if our goal is to assess the improvement of software development processes, the ability to answers the following three questions should be regarded as a central concern for the measurement of organizational performance:

(1) Are software projects delivered on time?

(2) Are software projects delivered on budget?

(3) Are customers satisfied with the delivered software?

5.4.2 Perceived level of success

Using organizational performance as the only dimension of success can entail some adverse complications. These complications include the instabilities of performance advantages, the causal complexity surrounding performance, and the limitations of using data based on retrospective recall of informants (March and Sutton, 1997). Furthermore, the extent to which organizational members' perceptions of SPI success reflect organizational performance is unclear, as is the extent to which perceptions are influenced by the software organizations' standards. Besides, research on both individual and organizational learning indicates that items that are perceived to be important by the persons concerned will be paid more attention to than items perceived as tangential to these persons (van der Bent *et al*., 1999).

If organizational members' perceptions do not reflect organizational performance, then increases (or decreases) in performance will not necessarily be translated into increased (or decreased) levels of perceived SPI success. A decrease in the perceived level of success, for example, may occur either because the software organization's performance has decreased, or because the organization has not adequately managed the perceptions of its members. In the absence of data about the relationships between actual performance, perceived performance, and customer satisfaction, therefore, assessments of SPI success is a question of both organizational performance and the perceptions of the organization's members.

Thus, to get a fuller picture of SPI success, we include the software organizations' perceived level of success, in addition to measures of organizational performance, in our model. Hence, SPI success will be measured along two dimensions:

(1) Organizational performance (i.e. time, cost, and quality).

(2) The level of perceived SPI success.

This is similar to the approach used by Teng *et al.* (1998) to measure reengineering project success and the relationship between characteristics of reengineering projects and implementation success. They measured success based on two different perspectives: perceived level of success and goal fulfillment.

5.5 Facilitating Factors

In this section, we focus on the facilitating factors and their associations with the learning cycle and SPI success by articulating the theoretical justification and detailed conceptualization that underlie each of the facilitating factors. Based on these conceptualizations, we develop a number of hypotheses regarding the key factors for success in SPI. These hypotheses are statistically formulated and tested in *Section 8.4*, while the details of the operationalizations necessary for validating and testing the hypotheses are described in *Chapter 7*. Furthermore, *Figure 6.6* (*Section 6.7*) presents the conceptual research model underlying the quantitative, hypothesis testing part of this study

The basis for this work was an extensive literature review within the fields of organization development and change, organizational learning, quality management, and SPI combined with empirical studies in four of the case companies and a survey of SPI experts from both industry and academia (see *Figure 6.3* in *Section 6.2*). The combined result of these investigations was the identification of the six key facilitating factors for SPI success described in this section: business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge.

5.5.1 Business orientation

Attaining business objectives is the ultimate target for any change program (Beer *et al.*, 1990; Cummings and Worley, 2001) and the role of process orientation and process improvement has long been recognized as essential factors to achieve business excellence and competitive advantage (French, 1963; French and Bell, 1999). Consequently, a clearly defined SPI program driven by strategic business needs, have been suggested as a key factor for success in SPI (e.g. Grady, 1997; Kitchenham, 1996; McFeely, 1996; Pulford *et al.*, 1996; Zahran; 1998).

Therefore, we would expect that a successful SPI program is one, in which SPI goals and policies have been clearly aligned with business goals, identifying the nature and direction of any SPI actions. Also, senior managers are becoming increasingly aware that SPI can be the principal means for successfully bringing their products to the market, winning contracts, and running their internal operations if they are to be adaptive and to survive in complex and rapidly changing markets (Biró and Tully, 1999; Brodman and Johnson, 1995).

Before SPI can be applied as a useful business oriented strategy in a changing environment, there must be a balance between long-term and short-term planning. In TQM, this balance is called *hoshin* management and is the primary guide to implement improvements (Uchimaru *et al*., 1993). Normally, a long-term plan – strategic hoshin – points toward the place a company wants to be in three to five years, while a short-term plan – annual hoshin – includes detailed propositions that must be executed within a year (Zultner, 1993). Ideally then, well-selected improvement actions should be based on long-term company needs and at the same time being able to survive organizational changes.

However, most software organizations are facing increasingly turbulent and complex environments which requires a more improvisational approach to improvement that recognizes the need for a dramatically shorter time-frame between planning and action (Dybå, 2000b). A fundamental challenge in SPI is, thus, to define a planning horizon that is neither too long nor too short for the organization's business situation, and ensure that the action plan is driven by both current and future business needs.

A final concern is that SPI is generally viewed as distinct from "real work". This distinction is made particularly clear within the "best practice" paradigm to SPI where improvement is seen as the transmission of explicit, abstract knowledge from someone who knows to someone who does not. However, practice is central to understanding software development, and consequently, any abstractions detached from practice cannot be the basis of improving that same practice (Brown and Duguid, 1991; Orr, 1996).

Furthermore, learning theorists such as Lave and Wenger (1991; Wenger, 1998) and Cook and Brown (1999) have rejected context-free transfer models that isolate knowledge from practice. On the contrary, they developed a view of learning as social construction, putting knowledge back into the contexts in which it has meaning. Such a situated and improvisational approach is also close to Levi-Strauss's (1966) concept of *bricolage*: the ability to "make do with 'whatever is to hand'" (p. 17). In other words, we expect that successful business-oriented approaches to SPI align improvement activities to the real needs of the developers rather than just to the abstract expectations of corporate strategy.

Based on the preceding arguments, *we define business orientation as the extent to which SPI goals and actions are aligned with explicit and implicit business goals and strategies*. A clear business orientation thus legitimizes the SPI initiative throughout the software organization. While it has a relatively low impact on local knowing it may, nevertheless, help software teams articulate the knowledge created in local groups. Business orientation is especially important in justifying concepts for inclusion in the organization's memory, since concepts must be selected that help the organization achieve it's business goals. Therefore, a clear business orientation will also encourage better utilization of organizational knowledge and facilitate the collective interpretation of knowledge.

Hence, we argue that business orientation is a key factor to facilitate a successful SPI program. *Ceteris paribus*,

Hypothesis 1: *SPI success is positively associated with business orientation.*

5.5.2 *Involved leadership*

Major changes require leadership. Thus, a predominant theme in the quality management, business process reengineering, organizational learning, and SPI literature is, as we have seen, the importance of leadership commitment and involvement in the implementation of improvement actions. Such involvement is of paramount importance to SPI since top executives often are committed to the status quo (Hambrick *et al*., 1993), and also because they ultimately determine the priority of the organization's SPI program.

Creating a vision is considered a key element in most leadership frameworks (Bennis and Nanus, 1997; Hesselbein *et al*., 1996; Humphrey, 1997; Kotter, 1996), and those leading the organization must take an active role in describing a desired future and energize commitment to moving toward it. The word "commitment", however, is often misused in popular management literature, and frequently confused with "compliance". In contrast to the compliant person, the committed person doesn't only play by the "rules of the game." He or she is *responsible* for the game, and changes the rules if that is necessary to achieve the vision. Furthermore, committed persons truly *want* the vision, while sincerely compliant persons merely accept the vision.

The command and control hierarchy of traditional organizations only requires compliance. However, traditional views on leaders as "heroes", who set the direction, make key decisions, and energize the troops, are changing (Senge, 1990). Creating vision is not enough; for assimilated learning to occur, leadership at any organizational level must engage in hands-on implementation of the vision. Thus, for building software organizations with learning capabilities, we need involved leadership at all levels that are committed to learning and SPI, and that take responsibility for enrolling others. This position is at odds with the traditional view of leaders as a synonym for top managers. Rather, we concur with Senge *et al*. (1999), and view leadership as the interplay between local line leaders, internal networkers, and executive leaders.

The *local line leaders* are people with accountability for results and sufficient authority to undertake changes in the way that work is organized and conducted at their local level. Examples are software managers, project managers, and team leaders. "Local line leaders are vital because only they and their colleagues, not executives, can undertake meaningful organizational experiments to test the practical impact of new ideas and approaches." (Senge *et al*., 1999, p. 17).

This view of local line leaders is similar to Nonaka and Takeuchi's (1995) emphasis on the role of middle managers in their "middle-up-down" process for knowledge creation. Contrary to the more established view of ignoring middle managers (often resulting in considerable management resistance to the change program), they recognized that middle managers are at the center of knowledge management at the intersection of the vertical and horizontal flows of information within the company. Thus, they argued that "middle managers are the key to continuous innovation." (*ibid*., p. 127). In other words, middle managers can be seen as playing a key role in facilitating the process of organizational knowledge creation and, hence, SPI.

The *internal networkers* belong more to the informal social networks of the company than to the hierarchy (Morgan, 1997). They complement the local line leaders, and function as change agents, carrying ideas, support, and stories through the organization. Such informal group leaders may become as powerful an influence on their network or group as much as any other rule, regulation, or manager. Despite their passion for creating better results within their unit, the limitation of the local line leaders is that they often have limited contact beyond their unit. The internal networkers, on the other hand, have a major strength in their ability to move about the larger organization, to participate in and nurture broad networks of alliances, and to help local leaders by assisting them and putting them in contact with others who share their interests (Senge *et al*., 1999). Thus, an SPI program can hardly succeed without the internal networkers being enrolled.

Due to constant change and increasing external pressures, *executive leaders* have a more challenging role today than ever before. They have overall accountability for organizational performance but less ability to directly influence actual work processes. They may be corporate presidents, vice presidents, or directors, and they are vital to profound change through their efforts to create an organizational environment for continual innovation and knowledge creation. One consequence of the executive leader's financial accountability, however, is that they disallow proposed improvement activities, which requires time and resources, on the grounds that the financial returns cannot be demonstrated, or that the existing control system would be undermined. In fact, "all of the research findings about the importance of teamwork, collaboration, commitment, and involvement fall on deaf ears, because in the executive culture, those are not the important variables to consider." (Schein, 1996c, p. 238).

It is important to note, however, that each type of leadership can only function effectively when the others function effectively, and that creating a culture for learning and SPI is the product of dynamic interaction among all members of the organization. Thus, much like Senge *et al.*'s (1999) "three leaders", Schein (1996a, 1996c) observed that a prerequisite for organizational learning is a reconciliation of the built-in conflict between the complementary roles of executives, engineers, and operators in corporations.

Accordingly, *we define involved leadership as the extent to which leaders at all levels in the organization are genuinely committed to and actively participate in SPI.* Involved leadership is thus important for any SPI initiative. By involving themselves in the challenges of software development and allowing software teams to act autonomously, the organization's leadership facilitates local knowing. They have an important role in facilitating the generation of new knowledge by creating a context that prioritizes and encourages dialogue and collective reflection. Also, the degree of leadership involvement influences what is considered important for inclusion in organizational memory.

Hence, we argue that involved leadership is a key factor to facilitate a successful SPI program. *Ceteris paribus*,

Hypothesis 2: *SPI success is positively associated with involved leadership.*

5.5.3 Employee participation

Employee participation, and the way people are treated, has been noted as a crucial factor in organizational management and development ever since Elton Mayo's (1933, 1945) famous productivity studies at Western Electric's Hawthorne plant. The results of Mayo's studies started a revolution in management thinking, showing that even routine jobs can be improved if the workers are treated with respect. Furthermore, the Tavistock studies (e.g. Trist and Bamforth, 1951; Trist, 1981) reframed the view of organizational systems, proposing the *socio-technical system* (STS) in which the best match would be sought between the requirements of the interdependent social *and* technical system. Together, these studies showed that there were powerful alternatives to the pervasive views of Taylor's (1911) concept of scientific management and Weber's (2000) description of bureaucracy.

Since then, participation and involvement has been one of the most important foundations of organization development and change (Cummings and Worley, 2001; French and Bell, 1999). Participation is also one of the fundamental ideas of TQM, which is a constituent of the philosophy of *kaizen*, or continuous shopfloor-led improvement (Imai, 1986). Similarly, participation has always been a central goal and one of the pillars of organizational learning. For example, autonomous work groups (Trist, 1981), quality circles (Aune, 1985; Ishikawa, 1990), survey feedback (Baumgartel, 1959; Neff, 1966), quality of work life programs (Davis, 1977), search conferences (Emery and Purser, 1996), and cultural analysis (Denison and Spreitzer, 1991; Schein, 1992) are all predicated on the belief that increased participation will lead to better solutions and enhanced organizational problem-solving capability.

There are several approaches to participation in systems design and development (see Gjersvik, 1993 for an overview), and in planned organizational change and learning (e.g. Greenwood and Levin, 1998; Zajac and Bruhn, 1999) that are relevant for SPI. The STS approach is, however, the most extensive body of scientific and applied work underlying employee participation today, with techniques and design principles derived from extensive action research in both public and private organizations across a diversity of national cultures. STS also has a particularly strong position in Scandinavia and the U.K., who has a long tradition for work-place democracy and participation. In fact, many of the early experiments of STS were pioneered in Norway (e.g. Thorsrud, 1968; Thorsrud *et al*., 1976).

An important aspect of STS and participation is "co-determination", i.e. the direct participation of workers in decisions about what should best be done at their own level. Within the context of software development, no one is more expert in the realities of a software company's business with respect to the day-to-day details of particular technologies, products, and markets than the software developers and their first-line managers are. Moreover, they are not only experts on how to do the work – they are also the experts on how to improve it. Therefore, it can be argued that the people are a software organization's most important source of productivity and profits. Hence, it is important to involve all those who are part of the problem or part of the solution, and have decisions made by those who are closest to the problem.

The STS theory on the efficacy of autonomous work groups is based on the principle of the cybernetic concept of self-regulation: The more the key variance can be controlled by the group, the better the results and the higher the member satisfaction (Trist, 1981). A consequence of introducing autonomous groups is a radical shift in the role of supervision. In contrast to the bureaucratic theory of control, the function of supervision in the STS approach is to manage the boundary conditions in the group's environment so that the group itself may be freed to manage its own activities.

Like quality circles and communities of practice, autonomous work groups are systems of participatory learning processes. Within such groups, people share tacit knowledge and through dialogue they bring this knowledge to the surface. Furthermore, they engage in discussions that affirm or modify theories in use and they innovate new problem-solving routines while simultaneously managing the context. In other words, they engage not only in socialization, but also in experiential learning and organizational knowledge creation.

To guide their actions in a common direction of improvement, software organizations need some kind of basic structure that is common to its members. Consequently, creating strong organizational cultures, has been the focus of both management consultants (e.g. Hunt and Buzan, 1999; Peters and Waterman, 1982; Peters, 1988, 1992) and organizational researchers (e.g. Martin, 1992; Schein, 1992, 1996a). Common to these writings is the idea that all people in an organization have common interests, and that participation can help the organizational actors to pull in the same direction – toward improvement. However, rather than developing unnecessary structures of domination trying to create a common way of viewing the organization and its environment, participation should be intended to exploit the diversity existing between the autonomous groups, using it to create a "space of possibility" (Gjersvik, 1993) that can be utilized in SPI.

With SPI becoming increasingly important, and people increasingly recognized as the principal source of competitive advantage, software organizations need to encourage and support collaboration in work groups and project teams, and engage in organizational learning. Participation is, thus, fundamental for creativity and innovation, and for showing respect for the opinions of others and for the belief that learning and competitive advantage can result from an organizational climate that cultivates a diversity of ideas and opinions. Therefore, participation should be offered and managed in such a way as to allow all employees to improve their work and to feel a sense of contribution to the organization and its missions. In our view, then, SPI is neither top-down nor bottom-up – rather, it is participative at all levels.

Hence, *we define employee participation as the extent to which employees use their knowledge and experience to decide, act, and take responsibility for SPI.* Therefore, employee participation is the cornerstone of our model. It is important for all the knowledge creating activities in the learning cycle. It is the basis for local knowing, since it is only through participation that collective action can be taken and tacit knowledge can be shared. Dialogue and collective reflection are meaningless concepts without participation and it is an important facilitator for the generation of valid organizational knowledge. Likewise, it is through collective processes of sensemaking and active participation that organizational knowledge is diffused and brought to use in new situations.

Consequently, we argue that employee participation is a key factor to facilitate a successful SPI program. *Ceteris paribus*,

Hypothesis 3: *SPI success is positively associated with employee participation.*

5.5.4 Concern for measurement

Software measurement is widely recognized as an essential part of understanding, controlling, monitoring, predicting, and evaluating software development and maintenance projects (DeMarco, 1982; Gaffney *et al*., 1995; Gilb, 1976; Grady and Caswell, 1987; Fenton and Pfleeger, 1996; Fenton and Neil, 1999; Möller and Paulish, 1993; Oman and Pfleeger, 1997) and as a necessary part of any SPI or change program (Basili and Rombach, 1988; Dybå, 2000e; Grady, 1992, 1997; Humphrey, 1989; Kitchenham, 1996; Pulford *et al*., 1996). This position is by no means unique to the software community. Measurement, and in particular; measurement of customer satisfaction, is at the heart of quality management and is also a major concern for related disciplines such as organization development (Cummings and

Worley, 2001), organizational learning (Argyris and Schön, 1996), strategic planning (Kaplan and Norton, 1996, 2000), and business process reengineering (Davenport, 1993; Hammer and Champy, 1993; Hammer, 1996).

The use of goal-oriented measurement (e.g. GQM) administered by the developers themselves to document and improve their own performance has been identified as an important source of improvement. In addition, it has been reported that external evaluations and assessments also improve performance (see *Chapter 2*). This crucial role of measurement is also recognized in the recent revision of ISO 9000: 2000 and in the guidelines for process improvement in the ISO/IEC standard on software process assessment (ISO/IEC WD 15504-4: 2001). Furthermore, it seems that these measurement aspects can complement each other, and that goal-oriented measurement can be integrated with assessment to form a comprehensive measurement-based improvement strategy (Jansen and Sanders, 1998; McFeeley, 1996; Pulford *et al*., 1996).

Measurement and analysis is not only an efficient means for identifying, recommending, and evaluating process change, it can also be of crucial importance for assisting and guiding ongoing change. In other words, feedback to managers and employees about quality performance provides a means of learning and maintaining improvement-oriented behaviors (Nadler, 1977). A major concern for a measurement program, therefore, is to provide opportunities for developers to participate in analyzing, interpreting, and learning from the results of measurements and to identify concrete areas for improvement.

Basically, the quest for measurement stems from a need for an analytical and factual basis to decision-making and process improvement. Nevertheless, we should remember that software development is largely an intellectual and social activity (Fuggetta, 1999; Hohmann, 1997; Ould, 1996), and to interpret any measurement results accordingly. However, there is reason to believe that providing a constant flow of quality data from a variety of sources may lead to improved performance. This point was also emphasized by Grady (1992) who made the following analogy:

> [A] software project is like a train in a tunnel rushing toward a light. With knowledge of its speed and schedule, knowing that it is on the right track, we can be reasonably assured that it's daylight we see at the end of the tunnel. Without such quantifiable facts, it is just as likely that the light we see is another train (or disaster) rushing toward us on the same track (*ibid*., p. 217).

Hence, *we define concern for measurement as the extent to which the software organization collects and utilizes quality data to guide and assess the effects of SPI activities*. In addition to personal and collective experience, a concern for measurement is thus important in order to validate newly created knowledge and to ensure that gains have in fact been made. Most importantly, a concern for measurement facilitates local knowing by acting as a foundation for the collection, analysis, and feedback of data. Also, ongoing feedback as a group process is specifically important since it can be an effective tool for bringing about changes in the way work is done as well as in establishing causal relationships and generating new knowledge.

Consequently, we argue that concern for measurement is a key factor to facilitate a successful SPI program. *Ceteris paribus*,

Hypothesis 4: *SPI success is positively associated with concern for measurement*.

5.5.5 *Learning strategy*

A critical challenge facing software organizations is the dilemma of maintaining the capabilities of both efficiency and flexibility. This situation, which requires the management of both stability and change, has been described with a punctuated equilibrium model (Tushman and Romanelli, 1985; Lant and Mezias, 1992). From this perspective, organizations experience periods of *convergence* punctuated by fundamental *reorientations*. Too much stability within the organization can be dysfunctional and too much change and turbulence leads to difficulties for managers and developers to make sense of their environments. In other words, the process of learning "involves the creation and manipulation of this tension between constancy and change" (Fiol and Lyles, 1985, p. 805).

Thus, researchers and practitioners increasingly distinguish between the more modest, or evolutionary, efforts toward change and those that are more fundamental and, in a sense, revolutionary (see *Chapter 2*). For example, Argyris and Schön's (1996) distinction between the concepts of *single-loop* and *double-loop* learning, corresponds to Fiol and Lyles' (1985) *lower-level* and *higher-level* learning, or what Senge (1990) called the difference between *adaptive* and *generative* learning. Similarly, March (1991, 1999) made a distinction between the *exploitation* of old certainties and the *exploration* of new possibilities. Common to all of these constructs, is that the lower level involves improving existing behaviors and making progress toward stated goals, while the higher level requires questioning the appropriateness of the goals and recognizing the subjectivity of meaning.

Evolutionary learning strategies represent relatively small or incremental changes, in the organization's products, procedures, or services. They are new to the organization but reflect an adaptation or simple adjustment of existing practices, and their implementation rarely requires changes in organizational structures or processes. In contrast, radical learning strategies represent larger changes in organizational products, procedures, or services. They reflect broader shifts in perspective and reorientation of existing practices and often require major changes in organizational structures or processes to implement.

In other words, software organizations can engage in two broad kinds of learning strategies. They can engage in *exploitation* – the adoption and use of existing knowledge and experience, and they can engage in *exploration* – the search for new knowledge, either through imitation or innovation (see *Table 5.2*). Exploitation is associated with systematic reasoning, risk aversion, defining and measuring performance, and explicitly linking activities to these measures. Exploitation involves improving existing capabilities, processes, and technologies, as well as rationalizing and reducing costs. Exploitation, thus, legitimates refining, standardizing, routinizing, and elaborating established ideas, paradigms, technologies, heuristics, and knowledge.

In contrast, exploration is associated with complex search, innovation, variation, risk taking, relaxed control, loose discipline, and flexibility. Exploration involves learning through discovery and experimenting with ideas, paradigms, technologies, strategies, and knowledge in hope of finding new alternatives and untapped opportunities that are superior to the current practice. Furthermore, it encourages the flexibility and resilience needed to cope with turbulent environments that are lacking in exploitation.

Table 5.2: Exploitation versus exploration (Dybå, 2000b, p. 85).

Exploitation	**Exploration**
♦ Refinement, routinizing and elaboration of existing ideas, paradigms, technologies, processes, strategies, and knowledge.	♦ Experimentation with new ideas, paradigms, technologies, processes, strategies, and knowledge in order to find alternatives that improve on old ones.
♦ Provides incremental returns on knowledge and low risk of failure.	♦ Provides uncertain but potentially high returns on knowledge and carries significant risk of failure.
♦ Requires personnel who are skilled in existing technologies.	♦ Requires personnel who are skilled in emerging or innovative technologies.
♦ Can generate short-term improvement results.	♦ Often requires a long time horizon to generate improved results.

Finding a good balance between exploration and exploitation is a recurrent problem of theories of adaptation (Argyris and Schön, 1996; Levinthal and March, 1993; March, 1991, 1999), which talk about balancing search and action, variation and selection, change and stability, and diversity and unity. It is important to note, however, that exploitation and exploration are complementary modes of organizational learning, and that both forms are necessary for software organizations. A basic problem, therefore, is to engage in enough exploitation to ensure short-term results and, concurrently, to engage in exploration to ensure long-term survival.

Thus, to be viable in the short term, a software organization needs to be efficient – to be good at what it does, to be able to lever on its present skills and capabilities. This is achieved by accumulating experience into organizational memory in a limited number of activity domains, and by increasing proficiency through repeated practice and the formalization of task-related knowledge. However, to be viable in the longer run, a software organization also needs to be able to develop new capabilities, to absorb or create new knowledge and new technologies. This is a more "messy" process, and many attempts at discovery or experimentation outside the bounds of current competencies or the norms set by existing routines might well prove to be unrewarding. Still, some of the deviations from standard practice will be promising enough to point to new ideas and directions, which the organization can turn into usable knowledge.

In short, exploration cannot realize its occasional gains without exploitation of discoveries, and exploitation becomes obsolescent without exploration of new directions. Thus, a software organization that specializes in exploitation will eventually become better and better to use an increasingly obsolete technology, while an organization that specializes in exploration never will realize the advantages of its discoveries.

Based on the preceding discussion we contend that exploitation and exploration are linked in an enduring symbiosis, and that each form requires the other in order to contribute effectively to a software organization's survival and prosperity. Consequently, *we define learning strategy as the extent to which a software organization is engaged in the exploitation of existing knowledge and in the exploration of new knowledge*.

Exploitation of existing knowledge is closely tied to all the knowledge creating activities in the learning cycle. It facilitates local knowing by presenting a set of previously learned lessons that can be used in exploring the contingencies of the current setting. It is particularly important in facilitating the generation of new organizational knowledge since this involves the articulation and packaging of local knowledge and experience. Furthermore, before locally created knowledge is appropriated as part of the organization's memory it must be related to the existing knowledge. Also, as we have seen, the interpretation of knowledge necessarily involves a relation between new and existing knowledge.

Consequently, we argue that exploitation of existing knowledge is a key factor to facilitate a successful SPI program. *Ceteris paribus*,

Hypothesis 5: *SPI success is positively associated with exploitation of existing knowledge.*

Furthermore, exploration of new knowledge is particularly important in facilitating the collective interpretation of knowledge through exploratory experiments and prototyping. It is also the basis for local knowing by mixing together the contingency of the present situation with the lessons learned from prior experience. *Ceteris paribus*,

Hypothesis 6: *SPI success is positively associated with exploration of new knowledge.*

5.5.6 Joint contribution of facilitating factors

So far we have discussed the possible independent contributions of business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge to SPI success. However, dependent variables such as organizational performance or other success measures, like the one used in this study are rarely determined by one single independent variable. In addition to examining the independent contributions, therefore, we also want to examine the joint contribution of the independent variables to SPI success. *Ceteris paribus*,

Hypothesis 7: *The six independent variables of business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge will explain a large amount of the variance in SPI success.*

5.6 Chapter Summary

In this chapter, we have developed a dynamic model of SPI which embraces the foundations of the learning software organization described in the previous chapter. The model was grounded in our personal experience and observations in the case studies, related to the extensive literature review, and focused on the three research questions posed in the introductory chapter of the thesis. A critical element for developing the model was the integration of SPI activities with the real, situated nature of software development, and to focus on the role of certain facilitating factors in the diffusion of knowledge and experience within and between groups of software developers.

First, organizational context was described as an important element that imposes constraints and opportunities about what and how the organization can learn. Thus, two contextual variables were included in the model to capture the most influential sources of variation: environmental turbulence and organizational size. Then, we emphasized the importance of acknowledging that the learning process is a dynamic interplay between two primary dialectics: one between the local and organizational level, the other between generating and interpreting knowledge. Next, SPI success was described in terms of organizational performance and the software organization's perceived level of success. Finally, we described the theoretical justifications, detailed conceptualizations, and hypotheses for the key factors of success in SPI. These hypotheses are summarized in *Table 5.3*.

In the next chapter, we describe the justifications for and the assumptions behind the research approach used in this study, as well as the details of the methods used to develop the theory, operationalize the measures, collect the data, test the hypotheses, and answer the research questions. The hypotheses are statistically formulated and tested in *Chapter 8*, while the details of the empirical justifications and operationalizations necessary for testing the hypotheses are described in *Chapter 7*.

Table 5.3: Research hypotheses.

Hypothesis 1:	SPI success is positively associated with business orientation.
Hypothesis 2:	SPI success is positively associated with involved leadership.
Hypothesis 3:	SPI success is positively associated with employee participation.
Hypothesis 4:	SPI success is positively associated with the concern for measurement.
Hypothesis 5:	SPI success is positively associated with exploitation of existing knowledge.
Hypothesis 6:	SPI success is positively associated with exploration of new knowledge.
Hypothesis 7:	The six independent variables of business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge will explain a large amount of the variance in SPI success.

CHAPTER 6

Research Methodology

"*There is nothing so practical as a good theory.*"

– Kurt Lewin

Doing research, whether in SPI or any other field, involves two fundamental problems in the pursuit of knowledge, which are often referred to as the "essential problems of science". That is: "How do we know what we know?" and "How do we acquire knowledge?". These problems can be traced back to the ancient Greeks who considered the primary role of science was to turn *doxa* (that which was believed to be true) into *episteme* (that which was known to be true). On the other hand, the Sophists questioned how, and even if, this could be done. They asked whether it was possible to actually know that something was true. The argument since then has centered on whether knowledge can ever be "proven".

This chapter describes the justification for and the assumptions behind the research approach used in this study, as well as the details of the methods used to develop the theory, operationalize the measures, collect the data, test the hypotheses, and answer the research questions. In order to provide more perspectives on the phenomena being studied, a mixed-methods design was used. The main method was quantitative. However, qualitative methods were used in a secondary role to help in conceptualization, in developing the theory, to operationalize definitions of concepts, to extend and generalize the findings of the main method, and to explain cause-effect relationships.

6.1 Justification for the Methodology

As a relatively young academic discipline like software engineering matures, it is important to follow scientific concepts and methods in developing and testing theories about software processes and the approaches used to improve them. Software engineering has done an outstanding job of developing specific solutions to specific problems. However, since software engineering research is often more solution-driven than problem-focused, a careful analysis of the problems that practitioners face often reveals that what the researcher thinks is a major practical problem has little significance; whereas, the neglected problem often turns out to be important (Potts, 1993). Moreover, much of the research is mere advocacy rather than valid research, following a model characterized by "conceive an idea, analyze the idea, advocate the idea." (Glass, 1994, p. 44). Thus, there is a need for solid and relevant theory based on sound and rigorous empirical methods of both technological and organizational phenomena in order to improve software engineering processes.

In *Chapter 4*, we argued that software development is a social process and consequently, that software process improvement is a socially constructed learning process. Unlike natural processes, which occurs separate from our perceptions of it, social processes are constructs of human intention and consciousness. Similarly, all research is based on some underlying assumptions about what constitutes "valid" research and which research methods are appropriate. Therefore, prior to deciding which research method to use, it is important to know what these (sometimes hidden) assumptions are.

Furthermore, we would like to emphasize that *all research is a social construction*, since all research procedures necessarily involve individuals, their activities, and their institutions in its instantiation. Research, therefore, is a convention – related to societal norms, expectations, and values – which is used to engage in a search for knowledge and understanding. In the words of Lee (1999a):

> Research, in my view, consists of individual and group activity aimed at the construction of theories, whether they are theories about nature or theories about individuals, groups, organizations, and society. As such, theories are only theories: they are fictions; they are social constructions. They are not already 'there', waiting to be discovered; rather, we invent them (*ibid.*, p. 11).

Thus, we *define SPI research as either enhancing existing theory or building new theory in order to improve the practice of software development in organizations*. A significant element in this definition is that, unlike computer science, SPI involves *both* the software system *and* the organization. Hence, the software system and the organizational context must be studied, understood, and managed together, not separately.

Kuhn (1996) discussed the importance of "paradigm" in explaining research assumptions, arguing that paradigms are "universally recognized scientific achievements that for a time provide model problems and solutions to a community of practitioners." (*ibid.*, p. x). Rather than Kuhn's (1996) specialized notion, we use paradigm in the broader sense of Burrell and Morgan (1979), who characterized the term as a "commonality of perspective which binds the work of a group of theorists together." (*ibid.*, p. 23). We argue that the perception of "paradigm" is an important issue for the understanding of SPI research. Indeed, Guba and Lincoln (1994) maintained that "Questions of method are secondary to questions of paradigm, which we define as the basic belief system or worldview that guides the investigator, not only in choices of method but in ontologically and epistemologically fundamental ways." (*ibid.*, p. 105).

Thus, the basic beliefs that define a research paradigm can be summarized by the answers given to the three fundamental questions of *ontology*, *epistemology*, and *methodology*. Furthermore, these questions are interrelated, so the answer to one question will constrain how the others can be answered. The ontological question is about the form and nature of reality and, therefore, about what there is that can be known about it. For example, if a "real" world is assumed, then what can be known about it is "how things really are" and "how things really work." (*ibid.*, p.108). In this case, only questions that relate to matters of "real" existence and "real" actions are admissible. Other questions, such as those concerning value or moral, would fall outside the realm of legitimate scientific inquiry.

The epistemological question deals with the nature of the relationship between the researcher and what can be known. The answer to this question is constrained by the answer given to the ontological question. Thus, if a "real" world is assumed to exist, then the researcher must be objectively detached from the phenomenon under study and value free in order to discover "how things really are" and "how things really work".

The methodological question deals with how the researcher gains knowledge about the phenomenon under study. Again, the answer that can be given to this question is constrained by the answers already given by the first two questions. That is, if a "real" world is assumed to exist, and the researcher strives to be an "objective" observer in the pursuit of knowledge, then not just any methodology is appropriate. Thus, "the methodological question cannot be reduced to a question of methods; methods must be fitted to a predetermined methodology." (*ibid.*, p. 108).

Although paradigms deal with first principles, or "ultimates" (Denzin and Lincoln, 2000), the sets of answers given to the ontological, epistemological and methodological questions are always human constructions, subject to human error and social acceptance. They define the worldview of the researcher, but "These beliefs can never be established in terms of their ultimate truthfulness (*ibid.*, p. 157).

Much of the debate concerning the applicability of research methods originates in a polarization, or "paradigm war" between the *positivist paradigm* and the *interpretative paradigm*. We argue that it is of great importance for SPI research to understand the assumptions behind these paradigms. Broadly understood, positivism is a philosophy that argues for the application of the methods of the natural sciences to the social sciences and thereby presupposes the unity of science (Delanty, 1997). The interpretive approach, on the other hand, maintains that the methods of the natural sciences are inadequate to the study of social reality (Lee, 1991). Furthermore, the positivist paradigm underlies what is called quantitative methods, while the interpretative paradigm underlies qualitative methods (Tashakkori and Teddlie, 1998). Therefore, the debate between these two paradigms has sometimes been called the qualitative-quantitative debate (Reichardt and Rallis, 1994).

The ontological position adopted by the positivists is one of "naïve realism". It postulates that the world is an objectively given reality, and that the objects and structures of the world exist as empirical entities, on their own, independent of the observer's appreciation of them. This naïve realism is, however, often replaced with a more moderate form of "scientific realism" which holds that the objective world can only be known through models of the world. These models are not absolute – they can never be known with certainty – therefore, the job of science is to develop better models of the world. As such, the logical positivist belief in "confirmation" is often replaced by Popper's (1959) concept of "falsification" (Cook and Campbell, 1979).

The positivist epistemology is essentially based on the traditional approaches in the natural sciences. The investigator and the investigated "object" are assumed to be independent entities, and the investigator to be capable of studying the object without influencing it. Furthermore, values and biases are prevented from influencing outcomes, so long as the prescribed research procedures are rigorously followed (Guba and Lincoln, 1994). In this perspective, the growth of knowledge is a cumulative process in which new insights are added to existing ones and false hypotheses are eliminated in a search for regularities and causal relationships.

Typically, positivist research is based on quantitative methods, such as surveys and experiments aimed at deductively testing theories, which are often expressed as hypotheses in terms of independent and dependent variables. In practice, therefore, positivist research methods involve three interrelated sets of logic: the rules of formal logic, the rules of experimental and quasi-experimental design, and the rules of hypothetico-deductive logic (Lee, 1999a). In summary, positivism is a socially constructed world, populated with researchers whose shared beliefs include the interrelated logic provided by the answers given to the ontological, epistemological and methodological questions described above.

Apparently, the vast majority of SPI research is done from a positivist perspective. This is reflected by the large number of SPI papers dealing with technical issues (Consolini and Fonade, 1997; Fuggetta, 1999; Glass, 1994). This is not surprising since most software engineering researchers have their background in the computer science and engineering fields, two of the so-called "hard" disciplines. Indeed, it seems as a confirmation of Kaplan's (1964) Law of the Instrument: "Give a small boy a hammer, and he will find that everything he encounters needs pounding. It comes as no particular surprise to discover that a scientist formulates problems in a way which requires for their solution just those techniques in which he himself is especially skilled." (*ibid.*, p. 28).

Contrary to the positivist paradigm, the ontological position adopted by the interpretive paradigm is essentially relativistic, which implies that one must understand the world from the inside rather than the outside. The interpretivist holds that reality is a subjective construction of the mind, and that its form and content depends on the individual holding the construction. Thus, what is subjectively experienced as an objective reality exists only in the observer's mind. Such constructions are not more or less "true", in any absolute sense, but simply more or less informed and/or sophisticated (Guba and Lincoln, 1994). Besides, the process of interpretation is iterative, which means that the subjective constructions and their associated "realities" are alterable. As a result, it is hard for a researcher within the interpretive paradigm to generate objective knowledge.

As seen from the interpretive paradigm, the conventional distinction between ontology and epistemology disappears, since "The investigator and the object of investigation are assumed to be interactively linked so that the 'findings' are *literally created* as the investigation proceeds." (Guba and Lincoln, 1994, p. 111, emphasis in original). As such, the interpretive researcher uses himself or herself as the instrument of observation (Lee, 1999a).

From a methodological point of view, interpretivist research can be seen as taking a hermeneutical approach to the interpretation of human and organizational behavior. It is focused on understanding a phenomenon in its context. This view draws upon the notion of the hermeneutic circle: In order to understand the part, the inquirer must grasp the whole, and vice versa (Schwandt, 2000). Consequently, the interpretive researcher typically makes an in-depth inquiry of a particular case, a single entity, or a specific event using qualitative methods such as case study or action research. In summary then, interpretivism, no less than positivism, is itself a socially constructed world, populated with researchers whose shared beliefs include the concepts of humanly created meanings, with the researcher himself or herself as the instrument of observation in an intrinsically iterative process of interpretation.

Based on the preceding discussion, the positivist approach and the interpretive approach would appear to be in opposition. However, it is important to note that, regardless of the research paradigm, both the research product and the research process is socially constructed. Research, therefore, "is not an entity that has an existence independent of knowing subjects; research (even research in the natural sciences) is a human creation and social activity." (Lee, 1999a, p. 11).

Burrell and Morgan (1979) contended that the ontological, epistemological, and methodological assumptions upon which each paradigm is based are so contradictory that they are mutually exclusive. Several other authors have fueled the paradigm war with statements about their incompatibility. For example, Guba (1987) stated that one paradigm precludes the other "just as surely as the belief in a round world precludes belief in a flat one." (*ibid.*, p. 31), while Smith (1983) stated the incompatibility thesis as follows:

> One approach takes a subject-object position on the relationship to subject matter; the other takes a subject-subject position. One separates facts and values, while the other sees them as inextricably mixed. One searches for laws, and the other seeks understanding. These positions do not seem compatible (*ibid.*, p.12).

Opponents of "paradigm incommensurability" counter with arguments that, while the central assumptions of each paradigm are indeed incompatible, the paradigm boundaries are permeable (Hassard, 1988; Lee, 1991, 1999a). Furthermore, Gioia and Pitre (1990) argued that perspectives involving multiple paradigms open more windows through which to view a particular phenomenon, permitting a more comprehensive outlook:

> This stance implies that the provincialism that comes with paradigm confinement might instead be turned toward the production of more complete views of organizational phenomena via multiparadigm consideration (*ibid.*, p. 587).

Schultz and Hatch (1996) introduced the concept of interplay between paradigms. They argued that, apart from paradigm incommensurability, there are two paradigmatic positions that can be used for multiparadigm research: (1) *paradigm integration* synthesizes contributions from different paradigms in an attempt to achieve a more general model or theory; while (2) *paradigm crossing* postulates interdependent relationships between paradigms, emphasizing interparadigmatic contrasts and connections.

Furthermore, Schultz and Hatch (1996) identified four basic approaches to paradigm crossing: *sequential*, *parallel*, *bridging*, and *interplay*. In sequential paradigm crossing the results of research in one paradigm are used to inform or advance research in another, while parallel paradigm crossing applies different paradigms at the same time to a particular problem. Bridging emphasizes similarities between paradigms at the expense of differences, while paradigm interplay simultaneously acknowledges both differences and similarities between paradigms.

Acknowledging the gap between the two major paradigms of organizational and information systems research, Lee (1991) reframed this gap as a "diversity" in methods whose effect is a mutually supported collaboration, rather than a widening separation, between the positivist and interpretive approaches. Using a sequential paradigm crossing approach, he proposed an integrated model of the positivist and interpretive approaches, in which he identified three levels of understanding: subjective understanding, interpretive understanding, and positivist understanding. *Figure 6.1* depicts the cyclical nature of these three levels of understanding. The *subjective understanding* consists of the everyday common sense and everyday meanings with which the human subjects see themselves and the organizational world around them, and which give rise to the behavior that they manifest in socially constructed settings. It provides the basis on which to develop the interpretive understanding.

The *interpretive understanding* belongs to the observing researcher and is the researcher's reading or interpretation of the subjective understanding. Lee (1991) made it clear that "a different reading or interpretation of what the organization means to the human subjects will lead to a different theoretical explanation for how the human subjects behave." (*ibid.*, p. 352). In other words, the interpretive understanding will influence our choice of theory and the modifications of this theory in the process of refining the research model. Once it is judged valid, the interpretive understanding provides the basis on which to develop the positivist understanding.

Like the interpretive understanding, the *positivist understanding* also belongs to the organizational researcher; it is the one that the researcher constructs and tests in order to explain the empirical reality being investigated. This explanation takes the form of formal propositions, and obeys the same rules of formal logic and controlled empirical testing that apply to scientific explanations in general. Lee (1991) emphasized that the construction of a positivist understanding without the aid of a careful interpretation of the subjective meanings would invite "ethnocentrism". That is, the application of the researcher's own subjective meanings that exist in his or her own culture or organization to the phenomenon under observation, instead of the subjective meanings that exist in the observed culture or organization.

In summary, the subjective understanding provides the basis to the interpretive understanding, which provides the basis to the positivist understanding, from which follow predictions about the subject's actions. Through the cyclical relationships between these understandings, there are three tests that the positivist understanding may undergo (Lee, 1991). First, the organizational researcher should ensure that the subjective meanings, earlier recorded in the interpretive understanding, have been built into the positivist understanding. Second, the organizational researcher should ensure that the actions, theorized by the positivist understanding, would be understandable to the observed subjects themselves, in terms of their own subjective understanding. Finally, the organizational researcher should compare the behavior that the positivist understanding predicts against the behavior that the researcher actually observes. In other words, the purpose of the final test is to confirm or disconfirm the positivist understanding through controlled empirical testing according to the rules of formal logic and the rules of hypothetico-deductive logic.

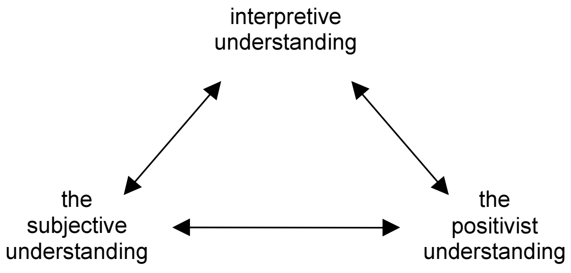

the interpretive understanding
the subjective understanding
the positivist understanding

Figure 6.1: Lee's (1991) model for integrating positivist and interpretive approaches.

Table 6.1: Comparison among the major research paradigms (adapted from Tashakkori and Teddlie, 1998, p. 23).

Paradigm	**Positivism**	**Interpretivism**	**Pragmatism**
Ontology	Naïve realism	Relativism	Accept external reality. Choose explanations that best produce desired outcomes.
Epistemology	Objective point of view. Knower and known are dualism.	Subjective point of view. Knower and known are inseparable.	Both objective and subjective points of view.
Methodology	Quantitative	Qualitative	Quantitative and qualitative
Logic	Deductive	Inductive	Deductive and inductive
Axiology	Inquiry is value-free.	Inquiry is value-bound.	Values play a large role in interpreting results.
Causal linkages	Real causes temporally precedent to or simultaneous with effect.	All entities simultane-ously shaping each other. It's impossible to distinguish causes from effects.	There may be causal relationships, but we will never be able to pin them down.

In a similar vein as Lee (1991), Goles and Hirschheim (2000), Lee (1999a), Neuman (2000), Tashakkori and Teddlie (1998), and Wicks and Freeman (1998) have argued for a pragmatist view and the use of mixed method research designs. In essence, pragmatists consider the research question to be more important than either the method they use or the worldview that is supposed to underlie the method, using the pragmatist credo of "what works" (Tashakkori and Teddlie, 1998). That is, *pragmatism* is based on the proposition that researchers should use "whatever philosophical and/or methodological approach [that] works for the particular research problem under study." (Tashakkori and Teddlie, 1998, p. 5). The overriding issue for the pragmatist is thus, whether or not something, be it philosophical assumptions, methodology, or whatever, is useful, with the term "useful" used in the sense that the something in question is instrumental in producing the desired or anticipated results. In order to provide a more comprehensive understanding of the pragmatist paradigm, we find it useful to compare the fundamental assumptions of pragmatism to those of positivism and interpretivism (see *Table 6.1*).

While positivists argue that there is an external, objective reality that exist independent of the individual, interpretivists counter that reality is equivocal, and that each individual uniquely interprets it. Pragmatists take the position that there is an objective reality, existing externally to the observer. However, this reality is grounded in the environment and experience of each individual, and can thus only be imperfectly understood. In a like vein, positivists believe that knowledge is objective, and is acquired by examining empirical evidence and testing hypotheses to uncover fundamental laws and causal relationships. By contrast, interpretivists believe knowledge is relative and reality too complex to be "known" by a single perspective and, furthermore, that all entities are simultaneously shaping each other in such a way that it is impossible to distinguish causes from effects. Pragmatists fall somewhere in between positivists and interpretivists. They view the process of acquiring knowledge as a continuum, rather than as two opposing and mutually exclusive poles of objectivity and subjectivity (Goles and Hirschheim, 2000).

Thus, the pragmatist view on the ontological and epistemological questions gives the pragmatist a choice of inductive and deductive logic in the course of conducting research on a question that needs to be answered (Tashakkori and Teddlie, 1998). Furthermore, it allows the pragmatist to select the methodology best suited to a particular research question, thus providing the conceptual foundation for the use of both quantitative and qualitative methods (Goles and Hirschheim, 2000).

Finally, positivists believe that research is value-free, while interpretivists, on the other hand, believe that research is value-bound. Once again, pragmatists take a middle position. They believe that values play a significant role in conducting research and in drawing conclusions from their studies, but they believe that the positivist zeal for objectivity is unattainable. Similarly, they see the interpretivist position as problematic due to its notion that all insights, perspectives, and values are equally valid (Wicks and Freeman, 1998). To the pragmatist, values are relevant and important only insofar as they influence what to study and how to study it. As Tashakkori and Teddlie (1998) stated:

> Thus, pragmatists decide what they want to research, guided by their personal value systems; that is, they study what they think is important to study. Then they study the topic in a way that is congruent with their value system, including variables and units of analysis that they feel are the most appropriate for finding an answer to their research question. They also conduct their studies in anticipation of results that are congruent with their value system (*ibid.*, pp. 26-27).

In accordance with the above discussion of the assumptions behind the major research paradigms, we argue that the practice of positivist research *can* be integrated with the practice of interpretive research. Furthermore, such integration is judged to be indispensable for valid and relevant SPI research. In our view, if SPI researchers are to get past the effects of the "paradigm war" and find meaningful ways to incorporate technological *and* organizational concerns into mainstream SPI research, we argue that a pragmatic approach applying mixed method research designs is required. Thus, pragmatism offers a practical basis for research in an applied field such as SPI: "Study what interests and is of value to you, study it in the different ways that you deem appropriate, and use the results in ways that can bring about positive consequences within your value system." (Tashakkori and Teddlie, 1998, p. 30). This does not mean however, that we sacrifice rigor for relevance. On the contrary, we argue for both – *both* relevant theory *and* rigorous empirical methods.

The *raison d'être* for a pragmatic approach to SPI research is to link theory and practice through mixed method designs and triangulation. *Triangulation* is a metaphor from navigation and military strategy that uses multiple reference points to locate an object's exact position (Jick, 1979). Given basic principles of geometry, multiple viewpoints allow for greater accuracy. Likewise, SPI researchers can improve the accuracy of their judgements by collecting different kinds of data bearing on the same phenomenon.

In the social sciences, the concept of triangulation can be traced back to Campbell and Fiske (1959). They argued that more than one method should be used to ensure that the variance in their research was accounted for by the trait under study, and not by the method that was employed. Denzin (1978) discussed four types of triangulation: data triangulation, investigator triangulation, theory triangulation, and methodological triangulation (see Patton, 1990, for extensive examples). Furthermore, Jick (1979) discussed “within-method” triangulation, which essentially involves cross-checking for internal consistency or reliability, and “between-method” triangulation, which tests the degree of external validity.

Based on the concept of triangulation, Creswell (1994) defined the following four types of mixed method designs, which have clear resemblance to Schultz and Hatch’s (1996) basic approaches for paradigm crossing:

- **Sequential**, or **two-phase designs**, where the researcher first conducts a qualitative phase of a study and then a separate quantitative phase, or vice versa.
- **Parallel/simultaneous designs**, where the researcher conducts the qualitative and quantitative phase at the same time and analyze the data in a complementary manner.
- **Equivalent status designs**, where the researcher conducts the study using both quantitative and the qualitative approaches about equally to understand the phenomenon under study.
- **Dominant – less dominant designs**, where the researcher conducts a study in which one paradigm and its methods are dominant, while a small component of the overall study is drawn from an alternative design.

In addition, Tashakkori and Teddlie (1998) defined a fifth type of mixed method design:

- **Designs with multilevel use of approaches**, where the researcher uses different types of methods at different levels of data aggregation.

Thus, mixed model studies can be defined as “*studies that are products of the pragmatist paradigm and that combine the qualitative and quantitative approaches within different phases of the research process*.” (*ibid*., p.19, emphasis in original).

The need for more pragmatism and mixed model studies has also been discussed within the software engineering domain. Potts (1993), for example, argued that “we must develop mixed qualitative and quantitative techniques, become more aware of professional guidelines about the interpretation of case-study and survey data, and reach a consensus about what constitutes acceptable evidence.” (*ibid*., p.27). Furthermore, with respect to research method, Jick (1979) noted that “the most prevalent attempts to use triangulation have been reflected in efforts to integrate fieldwork and survey methods.” (*ibid*., p. 604).

Following Jick (1979) and Lee’s (1991) model for integrating positivist and interpretive approaches, Gable (1994) demonstrated how qualitative case studies, when integrated with a quantitative survey, can be useful within information systems research: (1) as a source of rich detail to aid in the interpretation of quantitative findings from the survey; (2) as a further means of triangulation, by testing the propositions with the case sample as well as with the quantitative survey data; (3) to develop a close relationship with a few firms who may serve as the sample for pilot testing the survey instruments; (4) as a test of the contextual relevance of variables of interest; and (5) as an aid in identifying alternative *ex poste* models.

In the next section, we describe the sequential, mixed method research design used in this study, which applied the principles of pragmatism discussed above with Lee’s (1991) model for integrating positivist and interpretative approaches and Gable’s (1994) advice concerning the integration of case study and survey research methods.

6.2 *Research Design*

Generally, the selection of a research method for an empirical study depends on the variables studied, the current state of knowledge, and the objectives of the study (Galliers and Land, 1987). Also, practical limitations such as time, cost, and availability of data influence the choice. Furthermore, the choice of research design affects what concepts will be studied, how they will be measured, who (or what) will be studied, how the data will be obtained, and how it will be analyzed. Thus, the research design can be defined as a "blueprint" of research (Yin, 1994).

While it has been argued that "*Software engineering is a laboratory science*" (Basili, 1996, p. 443, emphasis in original), it is important to note that unlike the natural sciences, software engineering is a human based, social activity. The software engineering "laboratory" is, therefore, not a controlled environment like the laboratories of the natural sciences. Rather, our laboratory is the software industry – where practitioners build software systems (Potts, 1993). Therefore, in order to identify the factors for success in SPI, and given the aforementioned potential benefits of combining research methods within a single research design, we considered a combination of case study with survey research to be the most appropriate research design for this study.

The overall research design for the study is shown in *Figure 6.2*. It is a sequential, mixed method design (Creswell, 1994; Tashakkori and Teddlie, 1998), which builds on Scultz and Hatch's (1996) sequential paradigm crossing approach. It consists of four major stages: (1) a single pilot case study; (2) a multiple case study of twelve software organizations; (3) formulation of the model, and finally, (4) testing the hypotheses of the model by a survey.

The case study is a research method, which can involve a multitude of research strategies. Thus, several approaches of building theory from case study research have appeared in the literature, and much advice regarding the theory building process have been offered. However, in our process of inducting theory from the case studies, we based our design on the approaches offered by Eisenhardt (1989), Strauss and Corbin (1998), and Yin (1994).

The case study design included a single, exploratory, in-depth pilot case study (*Case A*: *Section 8.1*), followed by a more explanatory, cross-case analysis of twelve software organizations (*Case B* and *C*: *Section 8.2-8.3*; *Case D* and expert review: *Section 7.2*). Collectively, these case studies had a considerable influence on the model ultimately measured and tested by the survey. Ideas and concepts regarding SPI success identified in the exploratory pilot case study pointed toward important issues for further inquiry. The subsequent multiple case study had the objective of further exploring these issues, as well as identifying the key learning processes and factor for success in SPI.

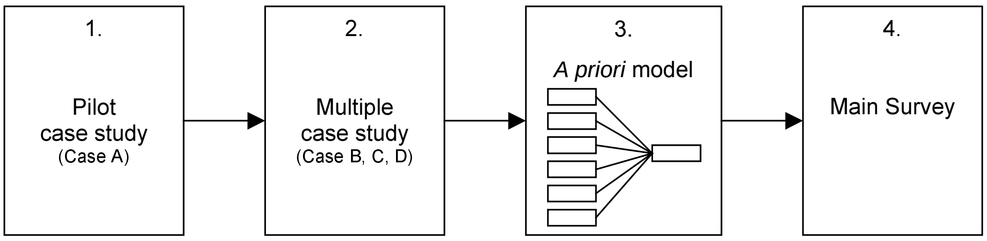


Figure 6.2: Overall research design.

Table 6.2: Characteristics of the pilot and multiple case studies.

Characteristic	**Pilot Case Study**	**Multiple Case Study**
Major objectives	Exploration	Exploration/explanation
Role of researcher	Participant/observer	Participant/observer
Description of cases	One software organization	Twelve software organization
Unit of analysis	Software organization	Software organization
Data sources	Interviews, observations, documents, forms	Interviews, observations, documents, questionnaires
Dependent variable	Not applicable	SPI success
Analysis methods	Description	Description/pattern analysis

Referring to the discussion on paradigm in the previous section, we argue that there must be a correspondence between the paradigmatic assumptions and the theory building effort. In this perspective, Gioia and Pitre (1990) defined theory as “any coherent description or explanation of observed or experienced phenomena” and theory building as “the process or cycle by which such representations are generated, tested and refined.” (*ibid.*, p. 587). Furthermore, they stressed that theory building should not necessarily be viewed as the search for the truth, but rather as comprehensiveness stemming from different worldviews.

While the goal of the positivist paradigm and, hence, for the quantitative part of this study, was to search for regularities and to establish cause-effect relationships, this was clearly not the goal of the qualitative part of the study. Rather, the goal for the qualitative part was to describe and explain in order to diagnose and understand. In other words, the goal of the theory building part of the study was “to generate descriptions, insights, and explanations of events so that the system of interpretations and meaning, and the structuring and organizing processes are revealed.” (*ibid.*, p. 588).

The theory building part of this study was, thus, based on data collection in case studies, and discovery through pattern analysis, which provided the possibility to identify relationships between concepts. Furthermore, the descriptions of the identified concepts and their relationships combined with an extensive literature review and prior experience provided the foundation for the developing the dynamic model of SPI, which was described in *Chapter 5*.

Using Gable’s (1994) classification, the theory building part of this study involved an exploratory, longitudinal, participant-observation single pilot case study (*Case A*) followed by an exploratory/explanatory, longitudinal, participant-observation, multiple case study of twelve software organizations (*Case B*, *C*, and *D*). *Table 6.2* indicates the main characteristics of the initial pilot case study and the subsequent multiple case studies with respect to major objectives, researcher role, number of cases, unit of analysis, data collection, dependent variable, and the primary method of analysis.

We would like to emphasize, however, that we did not act as an independent observer in these case studies. Rather, as part of a larger research team, we participated directly in the process of change within half of the organizations (e.g. Conradi and Dybå, 2001; Dybå and Moe, 1999), and more indirectly in the other half of the organizations (e.g. Dybå, 2000e; Stålhane *et al.*, 1998). This is also in line with our general view on the role of the SPI researcher: He or she should not be an objective disinterested scientist. On the contrary, *the SPI researcher should seek to actively contribute with his or her knowledge to the problem solving process in software organizations*. Our objective was, therefore, twofold: (1) to take action to solve a problem and (2) to contribute to a set of SPI concepts. Thus, the case studies could better be characterized as action research projects.

Generally, the researcher has less *a priori* knowledge of what the variables of interest will be and how they will be measured in a case study, compared with experiments and surveys (Benbasat *et al*., 1987; Gable, 1994). Typically, the researcher does not begin the study with a preconceived theory in mind. Rather, he or she begins with an area of study and allows the theory to emerge from the data (Strauss and Corbin, 1998). For that reason, and due to the exploratory and descriptive nature of the pilot case study, we did not specify an *a priori* dependent variable for the initial pilot case study. The following multiple case study, on the other hand, had the objective of identifying and describing the key learning processes and explaining, as far as possible, the relationships among the major variables studied as seen from the vantage point of the software organization. Thus, the case studies had a single unit of analysis – the software organization – and can therefore be characterized as "holistic" (Yin, 1994).

The dependent variable in the multiple case study was SPI success and the independent variables of interest were those factors which were posited to have a major influence on the level of SPI success. These factors were identified from the pilot case study, during the course of the multiple case study, from the literature and from our own prior experience as a consultant. Furthermore, the objective of the multiple case study was to identify the major contextual factors that could influence the relationships between the independent and dependent variables and explain, as far as possible, the moderating effects of these factors.

Multiple data collection methods were used both in the pilot case study and in the multiple case studies. Data were collected by means of qualitative techniques such as interviews, group discussions, feedback sessions, observations and archival sources, and quantitative techniques such as questionnaire surveys and various data collection forms according to pre-defined GQM measurement plans. Mintzberg (1979) described the synergy of combining qualitative and quantitative data in theory building as follows:

> For while systematic data create the foundation for our theories, it is the anecdotal data that enable us to do the building. Theory building seems to require rich description, the richness that comes from anecdote. We uncover all kinds of relationships in our hard data, but it is only through the use of this soft data that we are able to explain them (*ibid*., p. 587).

The main method of analysis for the multiple case study in *Case D* was pattern analysis, using the three basic types of coding procedures in grounded theory: open coding, axial coding, and selective coding (Strauss and Corbin, 1998). This analysis involved the identification of a pattern of variables from the literature and the pilot case study, and subsequently comparing this pattern with the patterns observed in the multiple case study. Based on this comparison, we developed an initial set of factors for SPI success with a corresponding set of hypotheses. Complementing these studies, we performed a review process with experts from academia and industry in order to derive a final set of factors and hypotheses (see *Figure 6.3*). The details of the literature review, which provides the *theoretical justifications* for the hypotheses, was presented in *Chapters 2-4*, while the details of the *empirical justifications* is presented in the next chapter.

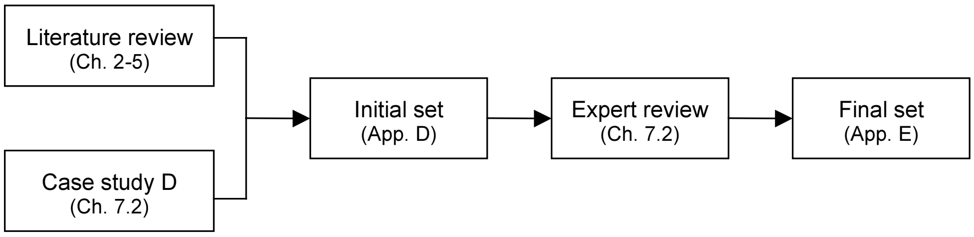


Figure 6.3: Process for developing and justifying the hypotheses.

The theoretical model and hypotheses developed in the theory building part of the study were tested using an explanatory, cross-sectional, ex post facto, field survey design. It involved several software organizations at one particular point in time, investigating to what extent the identified practices were implemented and what impact they had on SPI success. The unit of analysis, from which original data observations were obtained, was the software organization. Within this study, *a software organization is defined as a whole company or an independent business unit inside a larger company that has software development as its primary business*.

In this main part of the study, we used a mailed questionnaire as the data collection instrument. The results of reliability and validity analyses showed that the instrument has desirable psychometric properties as discussed in the next chapter.

The primary intent of the survey was, thus, to test the existence of statistical associations among the identified factors and SPI success, thus providing quantitative evidence for the key factors of success in SPI, and the generalizability of results. In addition to the independent and dependent variables, the contextual factors that might influence the results were included in the statistical analyses for greater quasi-experimental control. Thus the quantitative theory testing part of this study can be considered quasi-experimental (Cook and Campbell, 1979).

6.3 *Population and Sample*

Since this research is focused on an investigation of the importance of organizational issues in enabling SPI, software intensive organizations were considered as the target population. Furthermore, this population includes companies of different sizes, developing either software or combined software and hardware products for a wide variety of markets.

With respect to sampling, we approached the qualitative and the quantitative parts of the study differently. The primary goal of sampling in the quantitative part of the study was to get a representative sample from the population in order to make inferences from the smaller group about the larger group. For the qualitative part of the study we were less concerned about the representativeness or on detailed techniques for drawing a probability sample. Instead, we focused on studying specific cases and actions that could clarify and deepen our understanding. Furthermore, our major concern regarding the choice of the pilot case study was to seek a case that was recognized as a major success, thus deviating from the predominant characteristics of other cases.

Given these considerations, we established a mutual relationship with Nera AS – one of Norway's leading companies in the field of electronics – in the form of an ESSI PIE (ESSI project 21780, TELMET – Telecommunications Metrics Approach) in order to carry out the pilot case study. Nera provides an extensive range of monitoring and control systems for telecommunications networks – the TeleScada TMN. We studied and worked with a separate business unit within Nera that was responsible for the software product TeleScada TMN (now part of Protek Telsoft AS). Main customers are telecommunication service providers such as public, private, and military telecommunication organizations. Operations, processes, and procedures are regularly assessed by the British Standard Institute and the Norwegian Defense authorities, and are fully approved according to the military AQAP-110/-150 standards, the ISO 9001, and TickIT standards.

The overall goal of the PIE was to improve the quality and productivity of telecommunication software systems development by introducing metrics in the testing and inspection processes. In addition, the project acted as the first step in Nera's ambitious effort to advance in maturity from qualitative to quantitative goal setting. The experiment was, thus, used to test our ideas in a limited and controlled way, before proceeding with the implementation of an organisation wide measurement programme. For the purpose of the experiment, the **ami** method (Pulford *et al.*, 1996) and the GQM approach (Basili and Weiss, 1984; van Solingen and Berghout, 1999) were used to assist in planning, identification, and introduction of the metrics programme.

The PIE was organized by establishing a dedicated team within Nera, responsible for the TELMET project. The Software/Hardware Development department was represented with their manager in the metrics initiator and promoter roles. The Verification and Test group represented the System/Technology department, and the Customer Services department was represented with the product owner, who was responsible for refinement, improvement, and maintenance of products. When experimentation started, a person from the QA function joined the team and later became project manager for the TELMET PIE as requested by the manager of the Software/Hardware Development department. Finally, we were part of the team, with responsibilities for training and methodology support during all phases of the measurement program. For more details regarding the specific actions taken in the pilot case study (*Case A*), see *Section 8.1* and (Dybå *et al.*, 1997; Stokke and Palmstrøm, 1999; Stålhane *et al.*, 1997).

For the multiple case studies we sought cases that were actively pursuing SPI actions and, consequently, were concerned about succeeding with SPI. Thus, while the sampling procedure in the pilot case study can be characterized as "deviant case sampling" (Neuman, 2000; Yin, 1994), we applied "theoretical sampling" (Strauss and Corbin, 1998) to the multiple case study. That is, our growing theoretical interest in the key learning processes and facilitating factors for SPI success guided our selection of cases. Thus, we selected cases based on the new insights they could provide.

The multiple case studies were done within the context of the largest software process improvement effort to date in Norway, called SPIQ (Software Process Improvement for better Quality). The objective of SPIQ was to increase the competitiveness and profitability of Norwegian IT-industry through a systematic and continuos approach to process improvement. The goal of SPIQ was twofold: (1) to establish an environment for process improvement in the software organizations associated with SPIQ, and (2) to transfer and diffuse the knowledge gained to the remaining IT-industry in Norway through training, seminars, and conferences.

SPIQ was partly funded by the Norwegian Research Council and was managed by the IT-industry itself, with a consortium of twelve software companies actively participating with improvement projects, and an additional set of three to five companies participating in the internal experience forum's. The SPIQ member organizations covered companies of different sizes with a wide range of products and markets, developing either software or combined software and hardware products. Some companies belonged to the traditional electronics based IT-industry, while others belonged to the new dot-com industry. Besides SINTEF, the Norwegian University of Science and Technology, and the University of Oslo made up the research partners in the program. Results of *Case B* and *C* can be found in *Sections 8.2-8-3*, while results of *Case D* can be found in *Section 7.2*. Further details regarding the specific results of SPIQ can be found in (Arisholm *et al*., 1999; Dybå, 2000e; Dybå and Moe, 1999; Jørgensen *et al*., 1998; Lied and Stålhane, 1999; Stålhane *et al*., 1998; Westgaard and Gustad, 1999).

In the quantitative part of the study, software managers and quality managers in the Norwegian IT industry with corporate membership either in the Association of the Norwegian Software Industry (PROFF) or the Norwegian IT Technology Forum (ITUF) were chosen as the sampling frame for the survey. Both organizations are active in the area of software development and taken together, they were considered to be representative for software development within the Norwegian IT industry. Since the unit of analysis in this study was the software organization, the managers were asked to answer on behalf of their respective organizations.

Sample size might have a substantial impact on all results in quantitative research. Thus, an important consideration was the sample size needed for the survey. The effects of sample size are seen most directly in the *statistical power* of the significance testing and the *generalizability* of the results (Hair *et al*., 1998). Both issues were addressed in this study. At any given alpha level, the sample size will impact the power of the statistical test by either making it insensitive (at small sample sizes) or overly sensitive (at very large sample sizes).

Cohen (1988) examined power for most statistical inference tests and provided guidelines for acceptable levels of power, suggesting that studies be designed to achieve alpha levels of at least 0.05 with power levels of 80 percent. Using these guidelines, a sample size of about 100 to 120 respondents will suffice in testing the hypotheses and exploring the relationships in this study (see *Section 6.7* in this chapter for more details regarding these considerations).

The sample size affects the generalizability of the results by the ratio of observations to independent variables. A general rule is that the ratio should never fall below 5 to 1. However, the desired level is between 15 to 20 observations for each independent variable (Hair *et al*., 1998). When this level is reached, the results should be generalizable provided that the sample is representative. This study employs six independent variables. Thus, a sample size of 90 to 120 is needed to validate the generalizability of the results.

Sample size is also an important consideration in the discussion of the internal consistency of the measurement scales (a detailed examination of the internal consistency of the summated rating scales used in this study is made in the next chapter). The item analysis to choose a set of items that form an internally consistent scale, requires a sample size of about 100 to 200 respondents (Spector, 1992). Based on these guidelines, our target sample size for ensuring adequate statistical power, generalizability, and item analysis was a *minimum of 100 respondents*.

A random sample of 154 software and quality managers from the membership lists of the two associations were contacted by telephone to request participation in the study prior to mailing the questionnaires. All managers agreed to participate in the study. We provided the respondents with self-addressed, stamped return envelopes.

Table 6.3: Characteristics of the survey sample.

Characteristic	Frequency	Percent
I. Characteristics of the Respondent		
Average number of years worked in the company	8.4	
Average number of years worked with software development	11.4	
Highest completed education		
Bachelor's degree	38	31.7%
Master's degree	74	61.7%
Doctoral degree	5	4.2%
Other	3	2.5%
Job function		
Software manager	95	79.2%
Quality manager	25	20.8%
II. Characteristics of the Respondent's Company		
Number of software developers		
Less than or equal to 30	45	37.5%
Between 30 and 200	31	25.8%
More than or equal to 200	44	36.7%
Primary industry group		
Public sector	7	5.8%
Banking/finance/insurance	12	10.0%
Manufacturing	21	17.5%
IT sector	68	56.7%
Other	12	10.0%
Type of product business		
Standard applications (shelfware)	31	25.8%
Tailor made solutions for external customers	84	70.0%
Tailor made solutions for internal company customers	5	4.2%
Quality system in use		
Yes	86	71.7%
No	34	28.3%

Also, by keeping the questionnaire as short as possible (a pilot study of the survey questionnaire showed that respondents needed about 10 minutes to complete it), we combined several well-proven techniques for improving the response rate of mailed questionnaires (e.g. Fink and Kosecoff, 1998; Ilstad, 1997; Kanuk and Berenson, 1975; Neuman, 2000).

A total of 120 software and quality managers representing whole organizations or independent business units within 55 software companies completed and returned the questionnaire. This is within the limits discussed above for both adequate statistical power and generalizability of the results in this study. Furthermore, this represents an effective response rate of 77.9 percent, which is well within the norm of 60 +/- 20 percent for representatives of organizations and mid-level managers suggested for academic studies by

Baruch (1999), and considerably higher than prior mail surveys conducted in the Norwegian software and IT industry. As a comparison, a recent study by Larsen and Kautz (1997) on the status of quality assurance and software process improvement in Norway obtained a response rate of 13.3 percent. Stålhane *et al.*'s, (1997a) study of the customer's view of software product quality achieved a response rate of 8.4 percent, while Krogstie's (1996) study of the use of methods and CASE tools in Norway achieved a response rate of 14.9 percent. Given the high response rate in this study, no further analysis was done on the differences between respondents and non-respondents.

As shown in *Table 6.3*, the respondent's companies represent a wide variety of industry sectors. However, the sample is, as expected, biased in favor of IT companies (56.7% of the sample). To a large extent, the organizations develop software for external customers (approximately 96% of the sample). Furthermore, the sample shows a mix of both small and large organizations, with approximately one third (37.5%) of the organizations having 30 or less developers and approximately one third (36.7%) having 200 or more. A large majority of the respondent's organizations have a quality system in use (71.7%). The average length of the respondents' job tenure at their current organization is 8.4 years, while professional tenure (years in software development) is 11.4 years. Two thirds of the respondents (65.9%) holds a master's or doctoral degree.

6.4 *Variables and Measures*

Two processes of measurement were used in this study: conceptualization and operationalization. *Conceptualization* is the process of taking an abstract construct and refining it by giving it a conceptual or theoretical definition, while *operationalization* is the process of linking a conceptual definition to a specific set of measurement techniques or procedures (Neuman, 2000). This process, from abstract construct to concrete measure, for the independent and dependent variables in this study is illustrated in *Figure 6.4*. The process can, thus, be seen as linking the abstract (or theoretical) level together with the operational and empirical levels.

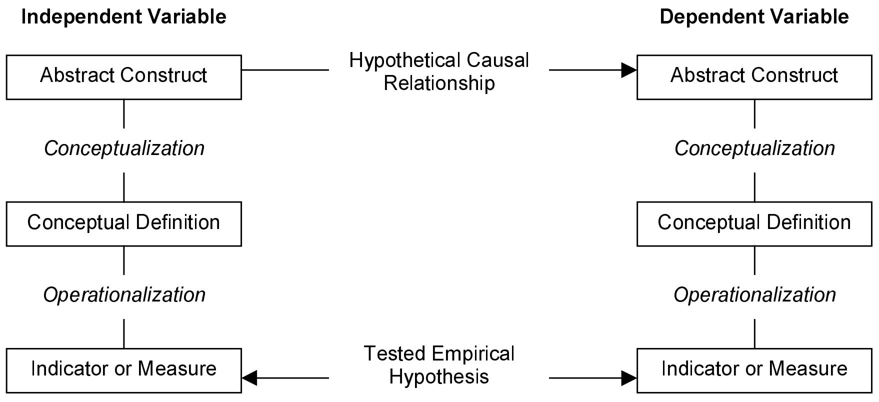


Figure 6.4: Conceptualization and operationalization (adapted from Neuman, 2000, p. 162).

The quantitative measurement process consisted of a relatively straightforward sequence of conceptualization followed by operationalization and data collection. Thus, constructs identified during the case studies were given clear, conceptual definitions (see *Chapter 5*). Next, these definitions were operationalized by developing a set of indicators for each concept (see *Chapter 7*). Finally, these indicators were applied to the measurement instrument used to collect the data. By contrast, the measurement process of the qualitative part of the study was not that clear cut. It was a more iterative process of forming and refining concepts grounded in the data and, thus, more integrated into the data collection process. Therefore, most of the conceptualization in the qualitative part of the study occurred *after* data collection, rather than prior to it.

In order to make sure that the concepts we developed were well grounded in the data as well as in the substantive theories related to the phenomenon under study, we combined the case studies with an extensive literature review and the use of additional experts from both academia and industry (see *Figure 6.3*). The data collected through these activities was subsequently used as input to the coding procedure in order to refine the abstract constructs and define the concepts. In this process, we applied the three coding procedures in grounded theory: open coding, axial coding, and selective coding (Strauss and Corbin, 1998) to the collected data.

During open coding, data were broken down into discrete parts, closely examined, and compared for similarities and differences. Data that was found to be conceptually similar or related was grouped under more abstract concepts or *categories* based on their ability to explain SPI success. This abstracting through comparative analysis was an important part of the conceptualization process, since it enabled us to reduce the number of concepts.

During the axial coding process, the identified categories were related to subcategories, or *indicators*, in order to form precise and complete explanations about SPI success. This step was important in building the theory and our goal was to systematically develop and relate categories to give them greater explanatory power.

Selective coding can be seen as the process of integrating and refining the theory (Strauss and Corbin, 1998). During this process, we integrated the data collected from the case studies in *Case D* (see *Appendix C* for an example) and related it do the prescriptions found in the literature (see *Chapters 2-4*), forming a final set of central categories (see *Appendix D*). This process resulted in the conceptual definitions of the independent and dependent variables described in the previous chapter. These definitions were then operationalized, and new data was collected to test the hypotheses in the quantitative part of the study. Thus, the grounded theory coding process may be regarded as *qualitative factor analysis* (Lee, 1999b).

The operationalization process in qualitative research significantly differs from quantitative research. Rather than turning a conceptual definition into a set of measurable indicators, operationalization in qualitative research is more of a detailed description of how the data was collected. As such it is an “after-the-fact” description more than a “before-the-fact” technique (Neuman, 2000). Consequently, we have described these details as part of the next section on data collection. Thus, the focus in the qualitative research was on conceptualization, while the quantitative research focused on operationalization. Details regarding the conceptual definitions were described in the previous chapter (*Chapter 5*) and detailed descriptions of the operationalization process is given in the next chapter (*Chapter 7*) on instrument development.

Based on the conceptual definitions of the central categories and the indicators identified in the qualitative part of the study, one dependent, six independent, and two moderating variables were operationalized and used in the quantitative part of this study to collect the data:

Independent variables. To measure the extent to which each of the six independent variables were practiced, we used multi-item, five-point, bipolar Likert scales that ranged from "strongly disagree" (1) to "strongly agree" (5) for all indicators. For each independent variable, the item ratings were summarized to form a summated rating scale for that variable. Thus, the operationalized levels of the independent variables were represented by the sums of the ratings of the multi-item scales. Furthermore, since this is the first study of its kind within SPI, all items were written specifically for this study based on the data collected in the case studies and through the literature review.

The reliability coefficients ranged from 0.78 to 0.87 (see *Table 7.4* in the next chapter). Since reliability coefficients of 0.7 or higher are considered being satisfactory (Cronbach, 1951; Nunnally and Bernstein, 1994), all scales developed for the independent variables used in this study were judged to be reliable. A more detailed analysis and discussion of the validity and reliability of the operationalized measures for the independent variables is presented in the next chapter.

Dependent variable. We operationalized and measured SPI success based on two multi-item measures. Each manager was asked to rate, on 5-point bipolar Likert scales, (1) the level of perceived SPI success and (2) the performance of their organization for the past three years with respect to cost reduction, cycle time reduction, and customer satisfaction. Two items were used to measure the level of perceived SPI success, while three items were used to measure organizational performance. As for the independent variables, all items were written specifically for this study. The ratings for the two performance dimensions were averaged to form a single measure of overall SPI success. The reliability coefficient for the five items of the dependent variable SPI success was 0.76.

Moderating variables. Two moderating variables – environmental conditions and organizational size – were operationalized and included in the hypothesis testing part of this study. Environmental conditions were measured using two semantic differential items written specifically for this study. Both items were rated on a seven-point, bipolar, adjectival, graphic scale. The two items were stable vs. unstable environment and predictable vs. unpredictable environment. The reliability coefficient of the summated rating scale was 0.80.

Organizational size was defined as the number of software developers in the software organization.

6.5 Data Collection Procedure

During the two-year period of the pilot case study (TELMET) – from April 1996 to April 1998 – and the three-year period of the multiple case study (SPIQ) – from January 1997 to December 1999 – we collected an extensive amount of both qualitative and quantitative data. Additional qualitative data was also collected during the period January 2000 to June 2001

(PROFIT) to aid in the interpretation of the quantitative results of the main survey. Data was collected by the means of conversational interviews, document studies, group discussions, measurement forms, and surveys.

Also, we developed a deeper understanding of the nature of SPI by being an active participant in several of the research projects covered by the case studies. Thus, the informal discussions, observations, and social relationships with developers and managers from these software organizations as well as with fellow researchers during these years have taught us a lot. Together with our prior consulting experience, this background has no doubt influenced the interpretation of important aspects of this research. In the same way, we have also contributed to the knowledge creating processes in the participating organizations. Thus, this research is part of a socially constructed setting.

The main data source in the pilot case study was several conversational interviews with the Vice President of Nera, the manager of TeleScada, the Software/Hardware department manager, the QA manager, the System/Technology department manager, the Customer Services department manager, and the software developers. Additionally, we were allowed to study all the documents at Nera that we thought relevant for this study. This included business strategies, project plans, quality manuals, internal and external assessment reports, product specifications, minutes of meetings, information leaflets, and job descriptions. Furthermore, as part of the TELMET team we participated in internal feedback sessions and group discussions, and had access to all measurement results of the participating Nera projects.

We grounded our own learning process about successful SPI actions in these data sources and together, they gave valuable input for reflection and to the process of forming the abstract constructs of this study. Most of the insights, however, came from the informal interviews, group discussions, and observations. One reason for this was that they were useful in shedding light on the importance of sensemaking and the relationship between espoused theories and theories-in-use. Another reason was that these activities allowed us to get to know more of Nera's organizational culture, contributing to a better understanding of the tacit assumptions underlying much of their actions.

The data collection methods used in the multiple case studies was much the same as in the pilot case study. However, in the multiple case studies we focused the data collection activities more directly toward learning processes and SPI success factors. With respect to SPI success factors, we performed four surveys among software managers, quality managers, software developers, and customer representatives in four of the case companies about the factors enabling SPI success (*Case D*). The data collection method used to derive these factors was three questionnaire items and follow-up group interviews (feedback sessions). For the purpose of the results presented in this chapter, data collection focused on answering one question:

> In your opinion, what are the three most important factors enabling SPI success in your organization?

In order to gain more background information about the enabling factors of SPI success, our questioning strategy included two additional questions. First, we asked the subjects about their most important argument *in favor* of SPI in their organization. Second, we asked the subjects about their most important argument *against* SPI in their organization. The answers to these questions gave us valuable information for interpreting the enabling factors (see *Appendix C* for an example).

In total, 54 managers, developers, and customer representatives answered the three questions and participated in the subsequent feedback sessions. Furthermore, as a complement to these four surveys and the accompanying literature review, we conducted an additional review process where we collected data from eleven SPI experts in both academia and industry. The review process consisted of three consecutive activities: (1) refinement of the derived categories, (2) prioritization of the hypothesized categories, and (3) prioritization of indicators within each category.

As for the success factors, we collected additional data regarding organizational learning processes and the transfer of experience among a further set of 23 software managers, quality managers, and software developers in five software organizations (*Case C*). The data collection method used was personal interviews focused on answering questions regarding (1) the familiarity with organizational routines; (2) the extent to which these routines were actually used; (3) the procedures for revising the routines; and (4) the organizational processes of transferring knowledge and experience. Furthermore, group reflection and discussion, and document studies were just as important for data collection in the multiple case study as it was in the pilot case study.

Three traditional methods of data collection were considered for the quantitative part of the study: mail questionnaire, telephone interview, and personal interview. Mailed surveys are usually self-administered, that is, the respondents fill in the questionnaires themselves, while telephone and personal interviews are administered by the researcher or another person. We evaluated the strengths and weaknesses of each of these data collection methods in order to arrive at the best solution for this study. Generally, the choice of methods depends upon several factors such as the availability of time and funds, the sample population, and the amount and kind of data to collect.

We chose a mail survey for this part of the study in order to cover a large number of software organizations in a wide geographical area with a fairly large number of questions within a reasonable period of time and cost budget. Additional advantages of mail surveys include anonymity and the lack of interviewer bias. On the other hand, the biggest disadvantage with mailed surveys is a potentially low response rate.

As explained in *Section 6.3*, we combined the mail survey with a telephone interview to request cooperation and participation in the survey prior to mailing the questionnaires, with several other well-proven techniques for improving the response rate. Furthermore, since mail surveys have been shown to be very effective, with high response rates for target populations that are well educated or have a strong interest in the topic (Neuman, 2000), we assumed that a fairly high response rate would be realistic for this study.

We scheduled the survey to the spring of 1999, and anticipated that the data collection period would last for about two months. Our major concern was, therefore, to have the survey completed before the beginning of the summer holiday season. The first questionnaires were mailed on 3 May 1999, and we continued mailing questionnaires for six weeks, until 10 June. The last response was received by 1 July 1999, thus, the data collection period for the main survey lasted for 9 weeks.

The questionnaire consisted of two parts: the first part asked for manager's ratings of their software department with respect to facilitating factors and SPI success; the second part asked for general background information and an assessment of the environment (see *Appendix F*).

In the first part, 38 separate items were used to measure the six facilitating factors of SPI success. However, two items were eventually deleted in order to improve the reliability of the final instrument. An additional set of five items was used to measure the SPI success construct. A complete list of these items appears in the next chapter. For each of the factors, a summated rating scale represented the level of practice as perceived by the managers. Thus, each manager generated seven scores; one for each of the facilitating factors, and one for the SPI success measure. Also, following Bagozzi (1996): "for scales comprised of the sums of items, about 4 to 8 items are usually adequate" (p. 41), all scales in this research had 5 to 8 items.

In the second part of the instrument, each manager assessed the environment using two semantic differential items, in addition to the general background and demographic information shown in *Table 6.3*.

In the next chapter, we detail the research method further and describe the design of the instrument used to collect the survey data and measure the key factors of success in SPI.

6.6 Data Analysis Techniques

In *Section 6.4* on variables and measures used, we also described the main, qualitative data analysis techniques used in this study: open coding, axial coding, and selective coding. In this section, we focus on the quantitative analysis techniques. Fundamentally, the quantitative data analysis involved two major steps: (1) describing the data – descriptive statistics – and (2) testing hypotheses and models – inferential statistics.

The primary purpose of the *descriptive statistics* was to describe and characterize the basic features of the sample under study in a manageable form. We provided simple summaries of the sample and the measures, which formed the basis of the quantitative analysis of the data. *Table 6.3* shows the major descriptive statistics for this study in terms of frequency distributions, percentages, and central tendencies. *Inferential statistics* was used to test the model and hypotheses in order to reach conclusions about the population parameters. Thus, while the descriptive statistics simply described the sample data, the inferential statistics allowed us to make inferences from the sample to the more general conditions of the population.

Most classical statistics make a number of assumptions about the properties (parameters) of the populations, such as the mean and standard deviation, from which samples are drawn. Generally, these *parametric tests* assume that the data being analyzed come from a normal distribution (Hays, 1994). *Non-parametric tests*, on the other hand, make no assumptions about the underlying distribution of the populations. For this reason, they are often referred to as distribution-free statistics (Davis, 1996). It is important to note, however, that although the non-parametric tests are distribution free, they still require that certain assumptions be met.

The fact that some tests exist in both parametric and non-parametric versions raises the question of when to use which version. In general, when the sample is normally distributed the parametric tests are more powerful than their non-parametric counterparts. As a consequence, we are less likely to commit an error of Type II when using a parametric test (see the next section for more details regarding error probabilities). There are, however, three situations when the use of non-parametric tests should be considered: (1) When the sample size is too small (usually < 30) for parametric tests; (2) when the data are not measured on an interval scale; and (3) when the data are not normally distributed.

With respect to (1) above, and as we have already described in *Section 6.3*, the sample size in this study was much greater than 30. Regarding (2), we will shortly justify and describe the techniques used in this study with reference to the debate regarding the meaningfulness of applying different classes of statistical methods to data arising from different kinds of measurement scales. Finally, referring to *Section 8.4* on empirical results, we describe the tests performed to examine whether the assumptions of the chosen tests are met.

Stevens's (1946, 1951, 1958, 1960) seminal work on measurement theory evoked considerable discussion and a great deal of controversy about the possible types of measurement scales and the permissible statistics for each scale. Stevens (1946) proposed that measurements fall into four major classes: *nominal*, *ordinal*, *interval*, and *ratio scale*, characterized by the type of transformations which map from one legitimate representation to another. Nowadays, these transformations are called *admissible* or *permissible* transformations (Hand, 1996). Furthermore, Stevens (1951) classified "permissible" statistical procedures according to these scales.

Thus, for nominal scales one-to-one transformations and summary statistics like number of cases, the mode, and contingency correlation would be permissible. For ordinal scales the permissible statistics included these plus the median, percentiles, and ordinal correlations, i.e. those that are preserved under monotonic increasing transformations. In addition, interval scales allowed means, standard deviations, and product moment correlations, since these statistics remain unchanged under linear transformations. Finally, ratio scales allowed all of these plus geometric means and coefficients of variation, which are unchanged by rescaling the data. *Table 6.4* provides an overview of Stevens' classification of measurement scales and the corresponding "permissible" statistics.

Whereas there is little dispute over whether nominal or ordinal properties have been established, there is often great dispute over whether or not a scale possesses a meaningful unit of measurement (Nunnally and Bernstein, 1994). Furthermore, there is dispute over whether the scale types should be used to proscribe the use of "inappropriate" statistical techniques (e.g. Briand *et al*., 1995, 1996b; Fenton, 1994; Fenton and Pfleeger, 1996; Hand, 1996; Tukey, 1961, 1962; Velleman and Wilkinson, 1993). Part of this dispute has to do with the definition of what is considered meaningful.

According to the representational point of view, the invariance definition is used as the single defining characteristic of meaningfulness. It should, thus, come as no surprise that practitioners come at odds with such a definition. Taking a representationalist position, Fenton and Pfleeger (1996) further advance this gap between theory and practice by stating that "Many software practitioners and researchers mistakenly think that to be meaningful, a measure must be useful, practical, worthwhile, or easy to collect. These characteristics are not part of meaningfulness." (*ibid*., p. 56).

Contrary to this view, and as we have already argued in *Section 6.1*, we take a pragmatist approach to SPI research with the aim of linking theory and practice. Thus, we oppose the representationalist position and argue that *the key to establishing measurement conventions about meaningfulness should rather be based on actual data and how well the scales work in practice*. This is also the essence of Velleman and Wilkinson's (1993) argument:

> Although we offer many arguments, the single unifying argument against proscribing statistics based on scale type is that *it does not work* (*ibid*., p. 68, emphasis added).

Table 6.4: Stevens' levels of measurement (adapted from Nunnally and Bernstein, 1994, p. 11).

Scale	Basic operation	Permissible transformations	Permissible statistics	Examples
Nominal	Equality vs. inequality	Any one-to-one	Numbers of cases, mode	Fault categories, industry sector
Ordinal	Greater than vs. less than	Monotonic increasing	Median, percentiles, order statistics	Maturity levels
Interval	Equality of intervals or differences	General linear $x' = ax + b$	Arithmetic mean, variance	Temperature (°C), student grades (?)
Ratio	Equality of ratios	Multiplicative $x' = ax$	Geometric mean	Temperature (K), length

Furthermore, the use of scale types to proscribe the use of "inappropriate" statistical techniques is controversial for at least two additional reasons (Briand *et al.*, 1996b). First, it is not always obvious that all statistics classified as non-parametric only make use of rank order information. Second, it is quite difficult to determine precisely the scale type of a measure. In fact, "Knowing the scale type of a measure with absolute certainty is out of the question in the vast majority of cases" (Briand *et al.*, 1996b, p. 73). However, although it cannot be demonstrated with absolute certainty that a particular scale is interval, the researcher may be confident that it is more than just ordinal. The question is, therefore, "If a scale is not an interval scale, must it be merely ordinal?" (Tukey, 1961, p. 245).

According to the representational position, the answer is that Stevens's (1946, 1951) scale types are absolute, and consequently, that data that are not fully interval scale must be demoted to ordinal scale. The pragmatic answer to this question, on the other hand, is that it is the *context* in which a statistic is used that should determine whether it is legitimate or not to promote a measure to a higher level. Tukey (1961) also noted the importance of meaning of the data in determining both scale type and appropriate analysis techniques. Furthermore, Wilkinson (1996) claimed that the popular advice of downgrading the measurement level of "sloppy data" to *ordinal* or *nominal* could only be considered a "parody of measurement theory." (*ibid.*, p. 487). In a similar vein, Velleman and Wilkinson (1993) stated that

> Approaches to statistics that starts from an a priori scale type and then proscribe the kinds of hypotheses that may be considered or the statistical methods and tests that may be computed based on that scale type are simply *bad science and bad data analysis* (*ibid.*, p. 70, emphasis added).

Even Stevens (1951) himself admitted that

> In the strictest propriety the ordinary statistics involving means and standard deviations ought not to be used with [ordinal] scales … On the other hand, … there can be invoked a kind of pragmatic sanction: in numerous instances it leads to fruitful results (*ibid.*, p. 26).

At the heart of this controversy, which has lasted for more than 50 years, is "The distinction, and confusion, between what is required to be able to *calculate* a statistic and what is required to *interpret* it." (Hand, 1996, p. 461, emphasis in original). While it is not our ambition to try to put an end to this debate, we maintain that an axiom system can be mapped to the world in more than one way and, hence, that *data analysis cannot be reduced to axiomatic mathematics*. Therefore, “An oversimplified and overpurified view of what measurements are like cannot be allowed to dictate how data is to be analyzed.” (Tukey, 1961, p. 247).

We must always remember that a model is nothing more than a limited abstraction of reality and not entirely accurate. A brilliant example of this point is made in René Magritte’s famous painting of a pipe – “La Trahison Des Images” – which also contains the words: “Ceci n’est pas une pipe“ (This is not a pipe). Magritte’s point was that however realistic a painting may be, it falls indefinitely short of being an actual pipe. Perhaps, in the spirit of Magritte, any model used in SPI should have in it the fundamental statement “This is not a process”.

Thus, in accordance with our pragmatic view on the use of research methods, we agree with Briand *et al*. (1995, 1996b) and Velleman and Wilkinson (1993) that the question under study rather than the data should determine the appropriate statistical technique:

> Scale type, as defined by Stevens, is not an attribute of the data, but rather depends upon the questions we intend to ask of the data and upon any additional information we may have (Velleman and Wilkinson, 1993, p. 69).

> In most cases, the questions to answer (i.e., our measurement goals) determine the scale under which data must be used, and not *vice versa* (Briand *et al*., 1996b, p. 74).

With the notable exception of Briand *et al*. (1995, 1996b), this pragmatic view on measurement is under-represented in software engineering due to the fact that the standard references on measurement theory in software engineering (e.g. Fenton, 1994; Fenton and Pfleeger, 1996) are only presenting the representational side of the debate. Incidentally, in the second edition of his classic book on *Psychometric Theory*, Nunnally (1978) characterized this view as the “fundamentalist” position.

As described in *Section 6.4* the scales for the facilitating factors of SPI and of SPI success are sums of individual Likert scale items. Furthermore, in the next chapter on instrument development we discuss a number of considerations regarding the selection of the number of points on the rating scales. Guided by these considerations, we constructed 5-point bipolar Likert scales for all questions in our final questionnaire. Although data from individual items are clearly ordinal, it is not equally clear what properties the summated scales have. However, we concur with the views of e.g. Nunnally (1978), Spector (1976, 1980), McIver and Carmines (1981), Hays (1994), Davis (1996), and Anastasi and Urbina (1997) and believe that it is permissible to treat a summated scale of individual Likert-scale items as leading to interval scale measurements that can be analyzed with parametric statistics.

A number of reasons account for this use of summated rating scales. First, empirical studies have found that Likert scales communicate interval properties to the respondents, and that they can be treated like approximate interval scales in empirical research work (Crask and Fox 1987; Schertzer and Kernan, 1985; Spector 1976, 1980). Also, Spector (1980) showed that people tend to treat categories in Likert scales as equidistant, regardless of the specific categories employed. Second, the use of summated rating scales, as leading to interval scale measurements that can be analyzed with parametric statistics is the established convention in the social sciences and in psychometric theory (Nunnally and Bernstein, 1994). Moreover, using parametric tests for scales that are not strictly interval does not lead, except in extreme cases, to wrong statistical decisions (Baroudi and Orlikowski, 1989; Briand *et al*., 1995, 1996b; Hays, 1994; Nunnally, 1978; Nunnally and Bernstein, 1994; Tukey, 1961, 1962; Velleman and Wilkinson, 1993).

Most results in this study are reported as either correlations or mean differences between groups. The most relevant parametric statistics for such analyses are product moment correlations, the *t*-test, and *F*, which are relatively robust with respect to measures that are "somewhere between" ordinal and interval levels of measurement, since they are little affected by monotonic transformations on variables (Nunnally and Bernstein, 1994). Furthermore, they are also rather robust to the violations of the normal distribution.

> Given the robustness and enhanced power provided by parametric tests, researchers are encouraged to use the parametric test most appropriate for their study and *resort to non-parametric procedures only in the rare case of extreme assumption violations* (Baroudi and Orlikowski, 1989, p. 98, emphasis added).

An interesting observation is also that Hays (1994), in the latest edition of his classic book, *Statistics*, removed the chapter on non-parametric methods. He made the following statement in the preface to the fifth edition to justify this:

> The remaining major change in the text itself is the elimination of the former Chapter 19, on other methods. These non-parametric techniques are still of interest and of value, but to me, they do not seem as critically important as they once did. The rather low power of many of the order methods and the growing evidence of the robustness of many of the traditional methods makes it seem less important to give the non-parametric methods as much space as heretofore (*ibid*., p. vi).

Since we have argued for the use of parametric statistics such as the Pearson product-moment correlation coefficient, the *t*-test, the *F* statistic, and multiple regression analysis to test the hypotheses in this study, it is essential that the assumptions of these analyses be met. Therefore, as shown in *Chapter 8* on research results, we examined the assumptions of normality, homoscedasticity, linearity, and independence of the error terms, as well as any potentially influential observations (outliers) and the degree of collinearity and multi-collinearity to validate the applicability of parametric statistics.

The results of these tests showed that all variables were within the acceptable range for the normal distribution assumption, and that the assumptions of homoscedasticity, linearity, and independence of the error terms were supported. Furthermore, no influential observations were identified, and the investigations of collinearity and multicollinearity indicated no problems. In other words, *there were no extreme violations to the basic assumptions underlying the chosen data analysis techniques* that could justify the use of less powerful non-parametric statistics. Lastly, the analysis of the quantitative survey data was conducted using the Statistical Package for the Social Sciences (SPSS 1999a, 1999b).

A final concern for the quantitative data analysis was to differentiate the successful from the less successful software organizations in order to explore further relationships. Thus, we defined contrasted criterion groups based on the upper and lower third of the performance distribution (see *Figure 6.5*). This is similar to the approach used by Hoch *et al*. (2000) to learn which factors make the difference between success and failure in the software industry at large. The advantage of this approach, is a much sharper differentiation between the contrasted criterion groups that otherwise would be made more diffused by borderline cases. The major disadvantage, however, is the exclusion of one third of the data points, and thus lower statistical power.

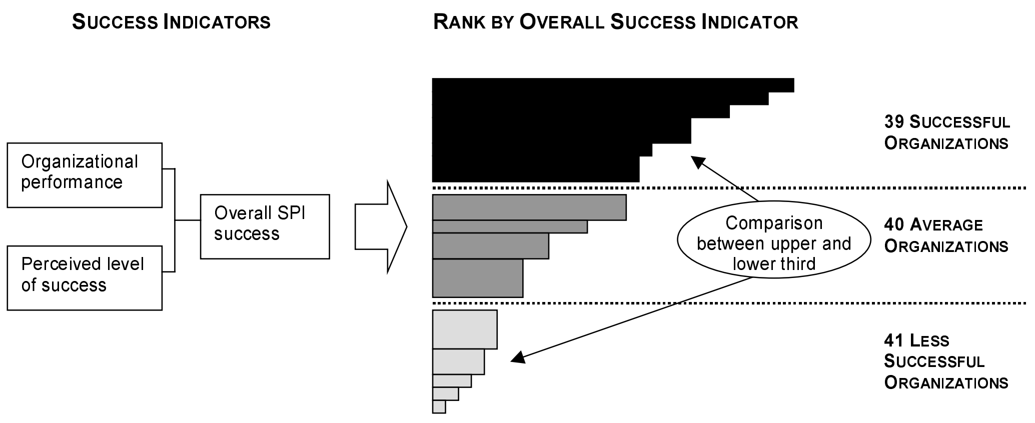

SUCCESS INDICATORS
RANK BY OVERALL SUCCESS INDICATOR
Organizational performance
Perceived level of success
Overall SPI success
39 SUCCESSFUL ORGANIZATIONS
40 AVERAGE ORGANIZATIONS
41 LESS SUCCESSFUL ORGANIZATIONS
Comparison between upper and lower third

Figure 6.5: Overall SPI success of top-third versus bottom-third software organizations.

6.7 Procedure for Hypothesis Testing

Apart from formulating the hypotheses, the most critical concern of the quantitative part of this study is the research model and procedure for evaluation of the hypotheses. Basically, such a procedure for hypothesis testing is a question of "How to Decide How to Decide" or "How Much Evidence Is Enough?" (Hays, 1994, p. 267). Consistent with the dynamic model of SPI defined in the previous chapter, *Figure 6.6* presents the conceptual research model underlying the quantitative, hypothesis testing part of this study. The model is comprised of the six independent variables identified in the previous chapter, the two moderating variables, and the dependent variable SPI success.

Hence, as the basis for testing the hypotheses, we have an empirically derived *model* of SPI success, as well as a validated and reliable *instrument* to measure the key factors of success in SPI. The model to be quantitatively tested by applying the measurement instrument can, therefore, be described as follows:

$$S = f(x_1, x_2, \cdots, x_n) \tag{6.1}$$

where S denotes the degree of SPI success, and x_i denotes the ith key factor for success. As already described in the previous chapter, we hypothesize that each identified x_i is positively associated with SPI success. We are, thus, not only able to identify the key factors for success in SPI, but also to measure the strength of their relationships with SPI success as well as their joint contribution and relative importance to the explanation of SPI success.

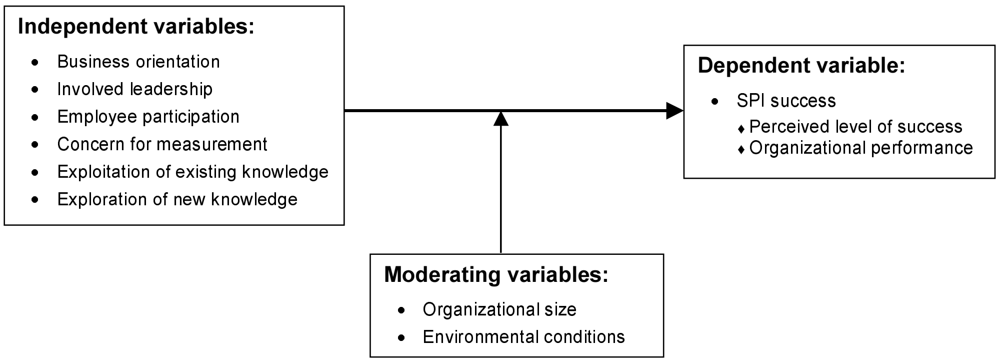


Figure 6.6: Conceptual research model for the quantitative part of the study.

We tested the hypotheses using the conventional logic of disconfirming hypotheses which is closely associated with Popper's (1959) idea of *falsification*, i.e. that claims to knowledge can never be proven or fully justified, but only be refused. Therefore, the starting point for significance testing of each of the hypotheses was a null hypothesis of no relationship between the variables being tested. Thus, if the null hypothesis is supported, then any propositions that there is a relationship must be rejected. In other words, we applied the most common approach of treating negative or disconfirming evidence for a hypothesis as more critical than positive or confirming evidence.

According to the Neyman-Pearson method of statistical inference, testing the hypotheses requires that we specify an acceptable level of statistical error, or the risk we are willing to take regarding the correctness of our decisions. Regardless of which decision rule we select, there are generally two ways of being correct and two ways of making an error in the choice between the null (H_0) and the alternate (H_A) hypotheses (see *Table 6.5*).

Relying on a weak test of a decision proposition can lead to a substantial money or opportunity loss for a software organization if the decision turns out to be wrong. In our experience, most software managers are more aware of the money losses than the opportunity losses and, as we argued in the previous chapter, thus committed to the status quo. The reason for this is simply that the costs of change are more visible than the potential return on the investment. Therefore, in this study we are most concerned about avoiding a Type I error, which is the error made when H_0 (the tested hypothesis) is wrongly rejected. In other words, a Type I error is committed whenever the sample results fall into the rejection region, even though H_0 is true. Conventionally, the probability of committing a Type I error is represented by the level of statistical significance, denoted by the lowercase Greek letter alpha (α). Conversely, the probability of being correct, given that H_0 is true is equal to $1 - \alpha$.

We tested the hypotheses using the conventional level of statistical significance established within the scientific community of $\alpha = 0.05$ as the decision rule (Baroudi and Orlikowski, 1989; Cohen, 1988; Miller *et al.*, 1997; Neuman, 2000; Sekaran, 1992), thus running a risk of 5% of incorrectly rejecting the null hypothesis. This means that correlations with $p > 0.05$ determine that there is no significant association between the variables being tested.

Table 6.5: Ways of being correct or making an error when choosing between two competing hypotheses (Hair *et al*., 1998, p. 11).

		Reality	
		H_0: No Difference	H_A: Difference
Statistical Decision	H_0: No Difference	1 - α	β Type II error
	H_A: Difference	α Type I error	1 - β Power

By specifying the level of Type I error, we also determine the probability of making an error of Type II, also known as beta (β), which is the probability of failing to reject the null hypothesis when it is actually false. Thus, when a sample result does not fall into the rejection region, even though some H_A is true, we are led to make a Type II error. Consequently, the probability of correctly rejecting the null hypothesis, i.e. the probability of making a correct decision given that H_A is true, must be 1 - β. We should note, however, that any change in a decision rule that makes the probability of one type of error smaller, usually make the other error probability larger.

The probability, 1 - β, is often called the *power* of the statistical test (Cohen, 1988). It is literally the probability of finding out that H_0 is wrong, given the decision rule and the true H_A. Therefore, it is important that the research design has adequate power to detect an anticipated effect size. Failure to provide an adequate level of statistical power has implications for both the execution and the outcomes of research. In the words of Baroudi and Orlikowski (1989): "If resources are limited and preclude attaining a satisfactory level of statistical power, the research is probably not worth the time, effort, and cost of inferential statistics." (*ibid*., p. 96).

Since we have chosen to protect ourselves more against false positives than false negatives, we set the power level to the usually recommended value of $1 - \beta = 0.8$ (Baroudi and Orlikowski, 1989; Cohen, 1988; Miller *et al*., 1997; Rademacher, 1999). I.e. the probability of not detecting an effect in testing the hypotheses in this study is 20%. Taken together with the conventional $\alpha = 0.05$, we thus have a β:α ratio of 4:1 of the two kinds of risks of being wrong in this study.

Furthermore, a hypothesis could be tested with a one- or two-tailed rejection region. Thus, another element to consider is the directionality or non-directionality of the statistical test. *Directional* or one-tailed hypotheses are implied when the basic question involves comparative terms like more than, increased, improved, declined, and so on. The essential question to be answered by the data has clear implications of a difference or change in a specific direction, if the result is to be called "significant". *Nondirectional* or two-tailed hypothesis testing, on the other hand, is called for when the question lacks a clearly defined notion of the direction of difference to expect if H_0 is false. Research questions would be without any specification of expected direction, e.g.: "Did something happen?", "Is there a difference?" or "Was there a change?" (Hays, 1994).

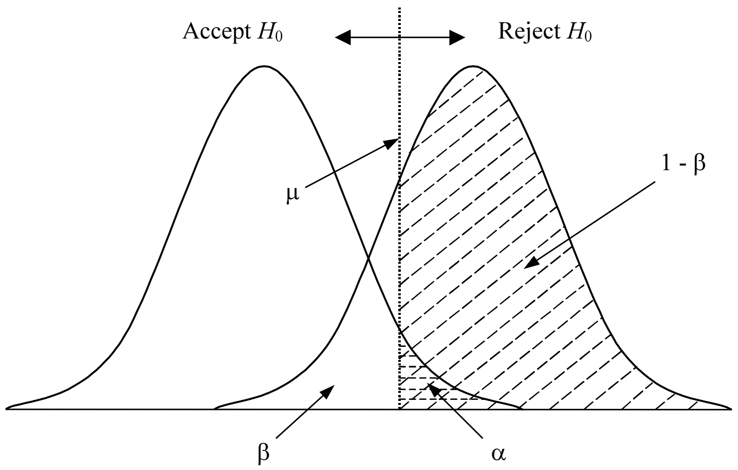


Figure 6.7: Statistical power and the probability of Type I and Type II error in testing a directional hypothesis.

The power of one-and two-tailed tests of the same hypothesis H_0 will tend to be different, given the same α level and the same true alternative H_A. For example, if a one-tailed test is used to test a mean and the true alternative μ_A actually is the direction of the rejection region, the one-tailed test is more powerful than the two-tailed over all such possibly true values of μ_A. Conversely, if the true alternative happens to be on the tail opposite the rejection region in a one-tailed test, the power is very low; in fact, here the power will be no more than α in the test of a mean (see *Figure 6.7*). Since all hypotheses in this study are directional, involving comparative terms like "positively associated with" we used one-tailed significance tests.

In addition to the directionality, and the type of the statistical test, the required power level ($1 - \beta$) is closely related to the significance criterion (α), the sample size (N), and the effect size (γ) (Briand *et al.*, 1996b; Cohen, 1988; Miller *et al.*, 1997). As we have already mentioned, power decreases with more restrictive alpha levels. With respect to effect size, a larger effect is more likely to be found than a smaller effect, and thus to impact the power of the statistical test. Finally, at any given level of alpha, increased sample size always produce greater power. However, increasing the sample can also produce "too much" power until at very large sample sizes almost any effect is significant (Hair *et al.*, 1998).

Having set the significance criterion to 0.05, the power level to 0.8 and decided to test the hypotheses with one-tailed rejection regions, what remains is to select the proper type of tests and a suitable sample size to ensure a meaningful study. In order to determine the N necessary for the specified power level and the significance criterion, we therefore have to estimate the effect size, which is the degree to which the phenomenon under study is present in the population. This is an essential concern, since "A recognition of the critical role played by effect size in the determination of the power of a statistical test is fundamental to adequate interpretation and application of research results." (Baroudi and Orlikowski, 1989, p. 90).

Furthermore, according to Baroudi and Orlikowski (1989), "effect size is probably the most important determinant of statistical power." (*ibid.*, p.90). However, estimating the effect size is a nontrivial task. Due to the limited number of empirical studies within software engineering and SPI, normative approaches to determine effect size are difficult to apply

(Miller *et al*., 1997). Therefore, we applied a judgmental approach based on Cohen's (1988) effect-size conventions of *small*, *medium*, or *large*. In information systems research and in the behavioral sciences, the operationalized definitions of the effect size for each of these categories have become something of a research standard for the most commonly used statistical tests (Baroudi and Orlikowski, 1989; Rademacher, 1999).

Cohen established these conventions in 1977 (Cohen, 1977), and they have been fixed since. His intent was that "medium [effect size] represent an effect likely to be visible to the naked eye of a careful observer ... small [effect size] to be noticeably smaller than medium but not so small as to be trivial, and ... large [effect size] to be the same distance above medium as small was below it." (Cohen, 1992, p. 156). *Table 6.6* gives the definition of the effect-size index and the corresponding operationalized effect size values for the four statistical tests used in this study.

In *Table 6.6*, the effect-size index, *d*, for the *t*-test of the difference between two independent means, is the difference expressed in units of the within-population standard deviation. Thus, an operationally defined medium difference between means is half a standard deviation. For testing the product-moment correlation of a sample for significance, the effect size is simply the population *r*. In the formula for the effect-size index, *w*, for the chi-square test for association in two-way contingency tables, *k* is the number of cells and P_{0i} and P_{Ai} are the null and alternate population proportions in cell *i*. Finally, the effect-size index f^2 is defined for either squared multiple or squared multiple partial correlations (R^2) (Cohen, 1988).

Having specified a significance criterion of $\alpha = 0.05$ and a power level of 0.80, we determined the necessary sample size, *N*, based on Cohen's (1992) tables, using a *medium* effect size as the dimensioning parameter. Furthermore, "the tabled sample sizes provide close approximations for one-sided tests at ½ α (e.g. the sample sizes tabled under $\alpha = .10$ may be used for one-sided tests at $\alpha = .05$)." (*ibid*., p. 156). *Table 6.7* shows the necessary *N* for power of 0.80 and significance criteria of 0.05 and 0.10 for the four tests used in this study.

Table 6.6: Effect-size indexes and their values for small, medium, and large effects for tests used in this study (adapted from Cohen, 1992, p. 157).

		Effect Size		
Statistical Test	**Effect-Size Index**	**Small**	**Medium**	**Large**
1. The *t*-test for the difference between two independent means	$d = \frac{m_A - m_B}{\sigma}$	.20	.50	.80
2. The *t*-test for the significance of a product-moment correlation coefficient, *r*	r	.10	.30	.50
3. The chi-square test for association in two-way contingency tables	$w = \sqrt{\sum_{i=1}^{k} \frac{(P_{Ai} - P_{0i})^2}{P_{0i}}}$	.10	.30	.50
4. Multiple and multiple partial correlation	$f^2 = \frac{R^2}{1 - R^2}$	.02	.15	.35

Table 6.7: N for small, medium, and large effect sizes at power = 0.80 for $\alpha = 0.05$ and 0.10 (adapted from Cohen, 1992, p. 158).

	α = 0.05			α = 0.10		
Statistical Test	**Small**	**Medium**	**Large**	**Small**	**Medium**	**Large**
1. Mean difference	393	64	26	310	50	20
2. Significance of *r*	783	<u>*85*</u>	28	617	<u>*68*</u>	22
3. Chi-square						
1*df*	785	87	26	618	69	25
3*df*	1090	<u>*121*</u>	44	880	98	35
5*df*	1293	143	51	1045	116	42
4. Multiple *R*						
$2k^a$	481	67	30			
$4k^a$	599	84	38			
$6k^a$	686	<u>*97*</u>	45			
$8k^a$	757	107	50			

[a]Number of independent variables.

Testing the hypotheses in this study involved significance testing of sample correlations in hypotheses 1 to 6, and significance testing of the multiple correlation coefficient in hypothesis 7. As seen from *Table 6.7*, a significance test of a sample r (i.e. hypotheses 1 to 6) at $\alpha = 0.05$ (one-sided), with a power level of 0.80, when the population r is medium ($r = 0.30$), requires a sample size $N = 68$. For a corresponding non-directional test, the required sample size would be $N = 85$.

In multiple regression, power refers to "the probability of detecting as statistically significant a level of R^2 or a regression coefficient at a specified significance level for a specific sample size." (Hair *et al.*, 1998, p. 165). With six independent variables (as in hypothesis 7) an alpha level of 0.05 and 80 percent power, *Table 6.7* shows that a sample of 97 respondents will detect a medium effect size for the F test of R^2 ($f^2 = 0.15$). Given the relationship between f^2 and R^2, this corresponds to an ability of detecting R^2 values of 0.13 or greater, and R values of 0.36 or greater.

In addition to testing the hypotheses, we also examined the effects of certain characteristics of the respondents and the respondents' organizations on SPI success using the t-test for the difference between two independent means and the Pearson chi-square tests of association. To detect a medium difference between two independent sample means ($d = 0.50$) with a directional test at $\alpha = 0.05$ (non-directional $\alpha = 0.10$) and power of 0.80 requires $N = 50$ in each group. To detect a medium degree of association in a contingency table with one degree of freedom in the population ($w = 0.30$) at $\alpha = 0.05$ and power of 0.80 requires $N = 87$. With three degrees of freedom, the corresponding sample size requirement becomes $N = 121$.

In summary, the results of the preceding power analysis have shown that a sample size of about 100 to 120 respondents is necessary for the conventional alpha level of 0.05 and 80 percent power to detect medium effect sizes in testing the hypotheses, and in exploring other potential relationships.

In agreement with the principle of falsification, we should speak of "*the credibility of a hypothesis relative to a given set of knowledge*" (Hempel, 1966, p. 45, emphasis in original), rather than trying to prove a hypothesis in the absolute sense. The diversity of evidence is an important factor in establishing such credibility and, thus, in the confirmation of a hypothesis. That is, the greater the variety of tests with favorable outcomes, the stronger the resulting support for the hypothesis. Consequently, we tested hypotheses 1 through 6 using both bivariate correlational and partial correlational analysis, as well as multiple regression analysis to simultaneously assess the degree and character of the relationships between each independent variable and the dependent measure. Hypothesis 7 was tested by multiple regression analysis both by simultaneously entering all independent variables to the equation and by applying stepwise regression. This way, we were able to determine not only individual relationships, but also the relative importance of each independent variable.

6.8 Chapter Summary

In this chapter, we have provided a comprehensive description of the justification for and the assumptions behind the research approach used in this study. Our main argument was that an applied discipline, such as SPI, requires a pragmatic research approach with mixed method research designs in order to link theory and practice and thus, ensuring both rigor and relevance. Then, based on a consideration of formal experiment, case study, and survey, we argued for an "industry-as-laboratory" approach, and described the specific research design employed in this study. The research design was a sequential, mixed method design, consisting of four major stages: (1) a single pilot case study (*Case A*); (2) a multiple case study of twelve software organizations (*Case B*, *C*, and *D*); (3) formulation of the model; and finally, (4) testing of the model by a survey. Next, the population and sample were described, followed by a presentation of how the variables and measures used emerged from the processes of conceptualization and operationalization. Data collection procedures and the techniques used for data analysis were presented next. Regarding the choice of appropriate test statistics, we emphasized the preeminence of the *research question* over considerations of either method or paradigm. Furthermore, *meaningfulness should be based on actual data and how well the scales work in practice*. Finally, we described the procedures used for testing the hypotheses, emphasizing the relationships between the significance criterion, statistical power level, effect size, and sample size.

The next chapter details the research method further and describes the design of the data collection instrument used to measure the key factors of success in the dynamic model of SPI (stage (3) of the research design). Specifically, the focus is on reliability and validity issues of the measurement scales. Then, in *Section 8.4*, the instrument is used to test the hypotheses and explore the relationships that were proposed in the model (stage (4) of the research problem).

CHAPTER 7

An Instrument for Measuring the Key Factors of Success in Software Process Improvement[1]

"It is the theory that decides what can be observed."

– Albert Einstein

In the preceding chapter we described the general research method used in this study. This chapter details the research method further and describes the operationalization and design of the instrument used to measure the key factors of success in SPI. The chapter provides a synthesis of the prescriptions for success found from the review of the quality management, organizational learning, and software process improvement literature discussed in *Chapters 2-5* of the thesis.

In addition to the theoretical justifications for the hypotheses presented in *Chapter 5*, this chapter provides a detailed empirical justification of these hypotheses in terms of exploratory studies in four of the case companies (*Case D*) and a survey among SPI experts from both academia and industry (see *Figure 6.3* in the previous chapter).

The main result from this chapter is an instrument, with satisfactory psychometric properties, for measuring the key factors of success in SPI based on data collected from 120 software organizations. In the next chapter, this instrument is used to test the hypotheses regarding these factors and SPI success, and to explore relationships between the success factors and the contextual variables.

[1] The results reported in this chapter are based on (Dybå, 2000a).

7.1 Introduction

During the last decade, the software industry has been more and more concerned about SPI. Consequently, we have witnessed a proliferation of models and initiatives all claiming to increase the likelihood of succeeding with SPI initiatives. The SPI literature is full of case studies and anecdotal evidence of successful companies and descriptions of their SPI programs. Several authors repeatedly discuss the importance of certain critical success factors (see e.g. *Chapter 2*). However, there has been no systematic attempt to synthesize and organize the prescriptions offered. The research efforts to date are limited and inconclusive and without adequate theoretical and psychometric justification. Even for commonly recognized factors such as management commitment and employee participation, no operational measures are available.

Despite an increasing focus on the nature and value of empirical software engineering (e.g. Basili *et al.*, 1986; Basili and Selby, 1991; Basili, 1996; Harrison *et al.*, 1999; Jeffery and Votta, 1999), poor theory development and inadequate measurement of constructs seems to be the norm for most studies regarding the key factors of success in SPI. The data collection device (in this study a questionnaire) used for measurement is commonly referred to as an *instrument* (Davis, 1996). However, instrument validation and reliability issues have been inadequately addressed in software engineering research, with only a handful of researchers (e.g. El Emam and Madhavji, 1995, 1996; Goldenson *et al.*, 1999a; Fusaro *et al.*, 1998; El Emam, 1998; El Emam and Birk, 2000) devoting serious attention to these issues.

Measurement instruments in both research and practice are expected to be valid and reliable (Straub, 1989; Anastasi and Urbina, 1997). The basic point is that users of a given instrument should obtain similar results. Thus, psychologists and psychometric theorists have developed rigorous methods for constructing reliable and valid instruments to measure variables in the social sciences (e.g. Cronbach, 1951; Likert, 1967; Nunnally, 1978; Nunnally and Bernstein, 1994; Anastasi and Urbina, 1997).

Hence, for the present research we define *instrument construction* as:

> the process of developing the data collection device in order to define and obtain relevant data for a given research question.

There are many reasons why researchers in SPI should pay closer attention to instrumentation. First, concerns about instrumentation are closely connected with concerns about rigor in empirical research methods for SPI. Second, greater attention to instrumentation permits confirmatory, follow-up research to use a tested instrument, hence, promoting cooperative research efforts (Hunter and Schmidt, 1990). Third, closer attention to instrumentation brings greater clarity to the formulation and interpretation of research questions (Straub, 1989). Finally, lack of validated measures in confirmatory research raises serious questions about the trustworthiness of the findings of the study.

In SPI research, as in social research and business research, we attempt to understand real-world phenomena through expressed relationships between research constructs (Blalock, 1969). However, these constructs are neither directly measurable nor observable, but are believed to be latent in the phenomenon under investigation. In SPI, for example, casual constructs like "organizational maturity" is thought to influence outcome constructs like "organizational performance". Neither of these constructs is directly observable, but relevant measures can be operationalized to serve as surrogates for these constructs.

Measurement of research constructs is neither simple nor straightforward. However, instrumentation techniques are available that allow us to construct research instruments that constitutes acceptable levels of reliability and validity. The process of developing the research instrument for this study was based on generally accepted psychometric principles of instrument design, and was carried out according to the nine steps shown in *Figure 7.1*.

The sections in this chapter are arranged according to these steps. *Section 7.2* presents the process of identifying the key factors of SPI success. The identification of items for each factor is described in *Section 7.3*, and the construction of measurement scales is presented in *Section 7.4*. In *Section 7.5*, we present a pilot test of the measurement scales and the overall instrument before data was collected (*Section 7.6*). Finally, *Sections 7.7*, *7.8*, and *7.9* are concerned with the reliability of the measurement scales, detailed item analysis, and validity of the measurement scales, respectively.

7.2 Key Factors of SPI Success

The starting point for developing the instrument was the extensive literature review of critical factors of quality management, organizational learning, and software process improvement discussed in *Chapters 2-4* and in *Section 5.5* (also see Dybå, 2000a). The reviewed authors have emphasized slightly different sets of organizational requirements for successful quality management and organizational change based on their personal judgment and experience. With a few exceptions (e.g. Saraph *et al.*, 1989; Powell, 1995; Ahire *et al.*, 1996; Black and Porter, 1996; El Emam *et al.*, 1998b) their requirements were not formulated on the basis of systematic empirical research.

As a complement to the literature study, we conducted an extensive exploratory study of factors enabling SPI success in four of the case companies (*Case D*). As we saw in *Section 6.5*, the data collection method used to derive key factors for success was three questionnaire items and follow-up group interviews (feedback sessions) in each company. For the purpose of the results presented in this chapter, data collection focused on answering one question:

> In your opinion, what are the three most important factors enabling SPI success in your organization?

In order to gain more background information about the enabling factors of SPI success, our questioning strategy included two additional questions. First, we asked the subjects about their most important argument *in favor* of SPI in their organization. Second, we asked the subjects about their most important argument *against* SPI in their organization. The answers to these questions gave us valuable information for interpreting the enabling factors (see *Appendix C* for an example of the results from one of the companies).

In total, 54 software managers, quality managers, software developers, and customer representatives answered the three questions and participated in the subsequent feedback sessions. The outcome of this study was a set of five enabling factors for SPI success: management commitment, participation, learning from experience, goal-/business orientation, and measurement. Each of these five factors was mentioned by at least three subjects in each organization.

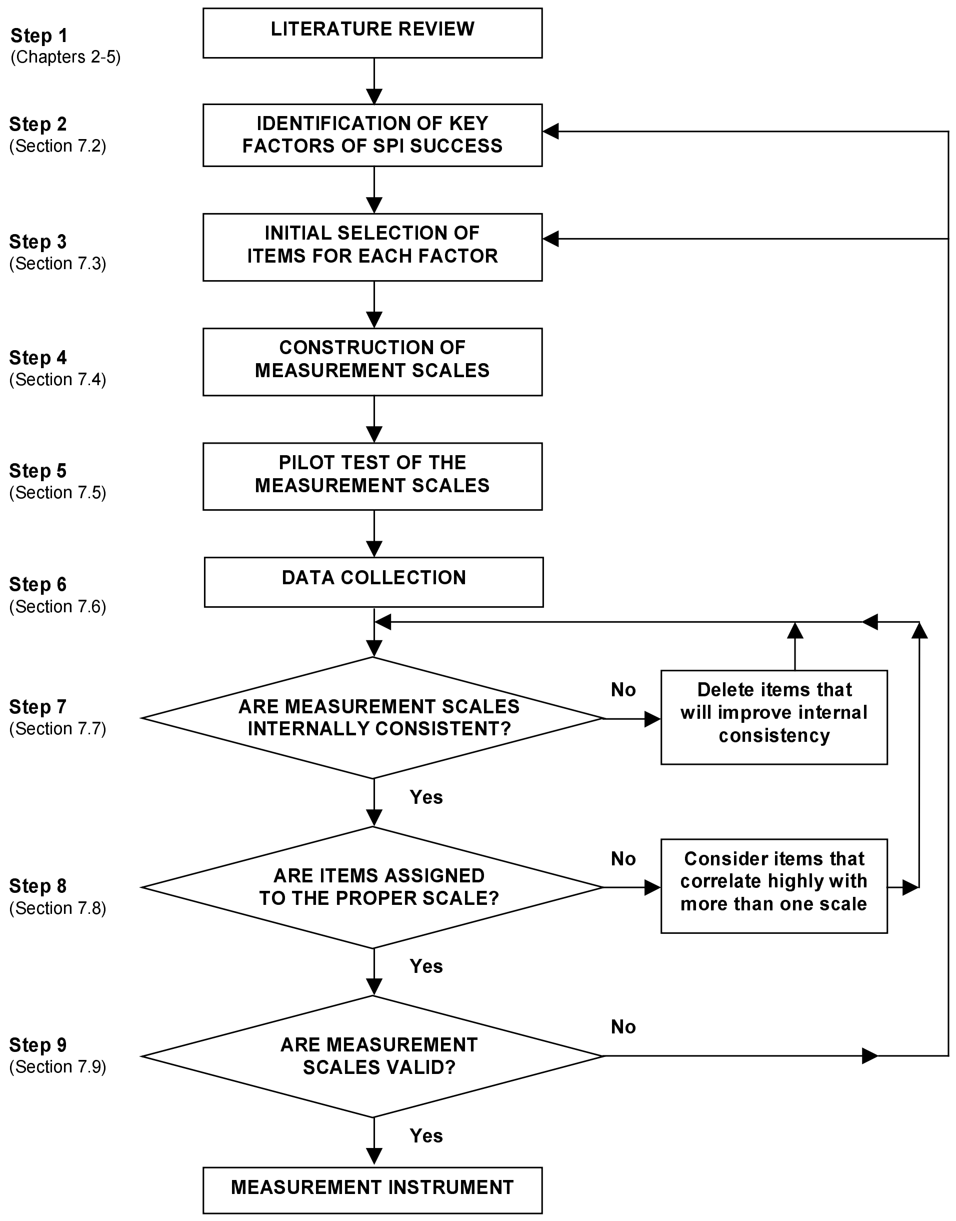

Step 1
(Chapters 2-5)
LITERATURE REVIEW
Step 2
(Section 7.2)
IDENTIFICATION OF KEY FACTORS OF SPI SUCCESS
Step 3
(Section 7.3)
INITIAL SELECTION OF ITEMS FOR EACH FACTOR
Step 4
(Section 7.4)
CONSTRUCTION OF MEASUREMENT SCALES
Step 5
(Section 7.5)
PILOT TEST OF THE MEASUREMENT SCALES
Step 6
(Section 7.6)
DATA COLLECTION
Step 7
(Section 7.7)
ARE MEASUREMENT SCALES INTERNALLY CONSISTENT?
No
Delete items that will improve internal consistency
Yes
Step 8
(Section 7.8)
ARE ITEMS ASSIGNED TO THE PROPER SCALE?
No
Consider items that correlate highly with more than one scale
Yes
Step 9
(Section 7.9)
ARE MEASUREMENT SCALES VALID?
No
Yes
MEASUREMENT INSTRUMENT

Figure 7.1: Instrument development process (adapted from Saraph *et al*., 1989).

Complementing the literature review and the exploratory study in the four software companies in *Case D*, we conducted a review process with eleven SPI experts in both academia and industry (see *Figure 6.3* in the previous chapter). The single most important criterion for choosing experts to this panel was “hands-on experience with SPI projects in the software industry”. Thus, we included software managers, senior software engineers, and researchers that actively participated *either* in the largest software process improvement effort to date in Norway, called SPIQ (Software Process Improvement for better Quality) (see Dybå, 2000e), *or* in a Process Improvement Experiment (PIE) within the European Systems and Software Initiative (ESSI).

The experts were, thus, chosen from companies of different sizes with a wide range of products and markets, developing either software or combined software and hardware products. Some of the industry expert's companies belonged to the traditional electronics based IT-industry, while others belonged to the new dot-com industry. Besides SINTEF, experts from academia were chosen from the Norwegian University of Science and Technology, and the University of Oslo, which made up the research partners in the SPIQ project. Of the panel of eleven experts seven had a Ph.D. in computer science, six had project management or SPI responsibility in industry, and five were researchers in software engineering and SPI in universities or research institutes.

The use of such an expert review process is common in instrument development, and the particular process used in this study was similar to that of El Emam and Madhavji (1996). The review process consisted of three consecutive activities: (1) refinement of the literature-derived prescriptions, (2) prioritization of the hypothesized categories, and (3) prioritization of indicators within each category.

Each expert received a memorandum with a brief summary of the prescriptions for success found from the literature review. Furthermore, as a result of this literature review, the experts were presented with an initial list of seven key factors for SPI success and a set of potential indicators for each factor based on the author's judgement (see *Appendix D*). Nearly all of the authors supported each of these factors; hence, together they were judged to define essential aspects of SPI.

Each expert was asked to (1) review the findings from the literature and the initial list of key factors, (2) delete the factors considered to have secondary or no importance, (3) add new factors considered to have high importance, and (4) give each of the *remaining* factors a priority code. The experts were also asked to add any literature references that, according to their judgement, was not represented by the review, i.e. that would add new prescriptions not covered by the presented literature review. Subsequently, the list of factors was modified, refined, and reworded during the course of the data collection.

A rank ordering was obtained using the total proportion of respondents who ranked a factor as critical for SPI success, as opposed to being of somewhat importance. During this step of the expert review process, it became clear that the experts placed similar levels of emphasis on the SPI success factors. The results of this survey and the subsequent rank ordering indicated which factors of SPI success which were considered most important, and which were considered less important.

During this process, it was clear that *the five factors that were ranked highest by the eleven SPI experts were the same factors that resulted from the exploratory study among the 54 managers and developers in the four software companies*. The two lowest ranking factors from the expert review process (experimentation and operational variety) were combined into one factor (exploration of new knowledge). Hence, the result of this investigation was the identification of six key facilitating factors for SPI success described and theoretically justified in *Chapter 5*. These six factors are listed in *Table 7.1*.

In the next section, we describe the identification of indicators for each of the facilitating factors of SPI success.

Table 7.1: Key factors of SPI success.

Key Factors of SPI Success	**Indicators for the Key Factors of SPI Success**
Business Orientation	♦ Extent of goal-orientation in SPI activities. ♦ Extent to which SPI goals and policy are understood. ♦ Degree of integrating SPI actions with “real” work. ♦ Extent to which SPI goals are aligned with business goals. ♦ Degree of balance between short-term and long-term goals.
Leadership Involvement	♦ Degree of management support to SPI activities. ♦ Acceptance of responsibility for SPI by management. ♦ Degree to which management considers SPI as a way to increase competitive advantage. ♦ Degree of participation by management in the SPI process. ♦ Amount of review of SPI issues in top management meetings.
Employee Participation	♦ Extent of employee involvement in decisions about what should best be done at their own level (co-determination). ♦ Extent to which employees contribute with improvement proposals. ♦ Extent of developer participation in the formalization of routines. ♦ Extent of on-going dialogue and discussion about software development. ♦ Extent to which employees have responsibility for SPI. ♦ Extent of developer participation in SPI goal setting. ♦ Extent of on-going dialogue and discussion about SPI. ♦ Extent of employee participation in development planning*.
Concern for Measurement	♦ Importance of measuring performance. ♦ Availability of quality data (defects, timeliness, etc). ♦ Extent to which quality data is available to developers. ♦ Extent to which quality data is available to managers. ♦ Extent to which quality data is used in SPI work. ♦ Amount of feedback provided to project teams on their performance.
Exploitation of existing knowledge (learning by experience)	♦ Extent to which existing knowledge is exploited. ♦ Extent of learning from past experience. ♦ Degree to which formal routines are based on past experience. ♦ Degree of systemization of past experience. ♦ Degree of internal experience transfer. ♦ Extent of risk aversion*.
Exploration of new knowledge (learning by experimentation)	♦ Degree of adaptability to rapid change, increasing complexity and environmental uncertainty. ♦ Extent to which innovation/creativity is encouraged. ♦ Extent of variety in the use of methods, techniques and tools. ♦ Degree of experimentation with new ideas, strategies, and technologies. ♦ Ability to question underlying values. ♦ Degree of flexibility in task execution. ♦ Degree of detail in task specifications (minimum critical specification). ♦ Importance of matching the variety and complexity of the organization’s environment.

* These items were eventually deleted to improve the reliability of the instrument.

7.3 *Item Creation*

Using the previously mentioned prescriptions found in the literature, several representative indicators were defined for each factor. In total, 59 indicators were initially defined for the original set of seven facilitating factors. These indicators are presented in *Appendix D*.

A process similar to that of identifying the key factors of SPI success was used to define the key indicators for each factor. Hence, for each factor, the eleven experts were asked to (1) delete indicators of secondary or no importance, or indicators that they considered inappropriate as descriptions of the factor under consideration, (2) add new indicators according to their judgement, and (3) give each of the *remaining* indicators a priority score.

As for the factors, a rank ordering was now obtained for the indicators, based on the experts' judgement. During this step of the expert review process it became clear that different experts placed different levels of emphasis on the indicators. The results of the subsequent rank ordering suggested which indicators that were considered more important or less important. Furthermore, according to Bagozzi (1996): "for scales comprised of the sums of items, about 4 to 8 items are usually adequate" (p. 41). Thus, following the expert review, the five to eight highest ranked indicators for each factor were chosen, for a total of 38 indicators. These indicators are shown in rank order within each factor in *Table 7.1*.

7.4 *Construction of Measurement Scales*

Based on the results from the preceding step, we defined one question for each indicator such that the theoretical abstraction of each indicator could be related more closely to everyday work situations. Furthermore, a subjective rating scale accompanied each question.

In selecting the number of points on a rating scale, Guilford (1954) suggested several considerations. If too few scale points are used, the answer scale is obviously coarse, and much information is lost because the scale does not capture the discriminatory powers that respondents are capable of making. Conversely, by adding too many scale points, the scale can become graded so finely that it is beyond respondents' limited powers of discrimination. Indeed, Miller (1956) argued that the average individual can process seven, plus or minus two (the "magical number") chunks of information at a time.

Likert and Roslow (1934) investigated the reliability of attitude scales by using three variations of the Likert scale. In addition to the original 5-point scale (Likert, 1932), they also used a 3-point and a 7-point scale. They concluded that the 5-point scale consistently yielded higher reliabilities than either of the two other scales.

More recently, Lissitz and Green (1975) conducted a Monte Carlo study of the effects of the number of scale points and homogeneity upon reliability. Their study showed that there was an increase in reliability as the number of points increased from two to five. However, reliability leveled off beyond five scale points. Consequently, Lissitz and Green (1975) concluded that since respondents are fallible, even fewer than five scale points may be necessary.

Qualitative studies by van de Ven and Ferry (1980) support the presented conclusions by Likert and Roslow (1934) and Lissitz and Green (1975). Guided by these studies, 5-point bipolar Likert scales were constructed for all questions in our final questionnaire. Responses were scored from 1 to 5, with a value of 1 indicating "Strongly disagree" and a value of 5 indicating "Strongly agree". Thus, we refer to each question and its associated 5-point scale as an *item*.

A typical item is presented below:

	Strongly disagree (1)	Disagree (2)	Undecided (3)	Agree (4)	Strongly agree (5)
Management is actively supporting process improvement activities.	☐	☐	☐	☐	☐

Random errors in single items will sometimes inflate and sometimes deflate the observed estimate of the true score. Therefore, when repeated measurements from the same respondents are taken over time, there will be inconsistencies in the observations. Consequently, single-item measures in questionnaires tend to be highly unreliable (Nunnally and Bernstein, 1994; Spector, 1992). On the other hand, when multiple items are combined into an estimate of the true score, errors tend to average out, leaving a more accurate and consistent (reliable) measurement over time (Spector, 1992).

Hence, to reliably measure complex concepts such as SPI success and facilitating factors, we developed multiple-item scales where more than one question was used to measure each concept. The actual level of practice for each facilitating factor is, thus, represented by the sum of the item ratings for that factor. We refer to these sums as *scale scores*. A vector of the scale scores for the six factors can thus be used to predict the software organization's chances of success with its SPI program.

As an example of a scale (see *Table 7.2*), we present the scale developed for the facilitating factor "Leadership Involvement" based on the five indicators resulting from the expert review process (the complete validated instrument is included in *Appendix E*).

Table 7.2: Leadership involvement scale.

Leadership Involvement	Strongly disagree (1)	Disagree (2)	Undecided (3)	Agree (4)	Strongly agree (5)
Management is actively supporting process improvement activities.	☐	☐	☐	☐	☐
Management accepts responsibility for process improvement.	☐	☐	☐	☐	☐
Management considers process improvement as a way to increase competitive advantage.	☐	☐	☐	☐	☐
Management participates actively in process improvement activities.	☐	☐	☐	☐	☐
SPI issues are often discussed in top management meetings.	☐	☐	☐	☐	☐

7.5 Pilot Test of Measurement Scales

The next step in the instrument development process was a pilot test of the measurement scales and of the overall instrument. In general, the pilot sample should be as similar to the target population as possible (Nunnally and Bernstein, 1994; Fink and Kosecoff, 1998). Since the primary objective was to develop an instrument to measure a software organization's score on the six facilitating factors of SPI success, first line software managers and quality managers were considered appropriate subjects (see *Sections 5.3.1* and *6.3*). Including both of these management groups are also important in order to have a more balanced view, since their attitudes may differ widely.

Furthermore, since this was a pilot test, the sample size was kept quite small. Questionnaires were distributed to a convenient sample of twelve managers in eight companies that either participated in the SPIQ program or were involved in an ESSI PIE. Each manager assessed his/her company by rating each measurement item using the scales described in the previous section. The managers mailed the completed questionnaires directly to the author to ensure confidentiality and anonymity of each response.

The goal of the pilot test was twofold: (1) to ensure that the mechanisms of compiling the questionnaire had been adequate, and (2) to make an initial reliability assessment of the measurement scales. The first aim of the pilot test was accomplished by having two professors of SPI research and one professor in psychological testing review the completed questionnaire, and comment on its length, wording, instructions, and format before it was sent out. Subsequently, each manager was asked to make the same kind of comments. None of the comments from this review implied a need to change the questionnaire.

Analyzing the correlation of items within each scale (item-item), the corrected item-to-total (item-scale) correlations, and the item standard deviation scores accomplished the second aim of the pilot test. Based on this analysis, we deleted one item in the "Employee participation" scale (marked with an asterisk in *Table 7.1*). This item was also the lowest ranked item in the previously described expert review of the "Employee participation" scale.

7.6 Subjects and Instrument Administration

Target respondents for the study were software managers and quality managers in Norwegian IT companies developing software. These managers were chosen since they serve as a bridge between the visions of top management and the often chaotic reality of the developers. Hence, they are the key knowledge engineers in their respective companies (Nonaka, 1991; Nonaka, 1994; Nonaka and Takeuchi, 1995). These managers were used in this study to answer on behalf of their respective organizations. Thus the unit of analysis, from which original data observations were obtained, was the software organization. Within this study, *a software organization is defined as a whole company or an independent business unit inside a larger company that has software development as its primary business.*

The objective was to develop an instrument that could be used in companies of different sizes, developing either software or combined software and hardware products. Therefore, managers were chosen from companies with corporate membership either in the Association of the Norwegian Software Industry (PROFF) or the Norwegian IT Technology Forum (ITUF). Taken together, these two organizations were considered to be representative for software development within the Norwegian IT industry.

Sample size is an important consideration in the discussion of the internal consistency of measurement scales. Thus, for the purpose of constructing a measurement instrument with satisfactory psychometric properties, the item analysis is the dimensioning factor. Generally, the item analysis to choose a set of items that form an internally consistent scale, requires a sample size of about 100 to 200 respondents (Spector, 1992). Based on these guidelines, our target sample size for ensuring adequate item analysis was a *minimum of 100 respondents*.

As we saw in *Section 6.3*, a random sample of 154 software and quality managers from the membership lists of the two associations were contacted by telephone to request participation in the study prior to mailing the questionnaires. A total of 120 software and quality managers representing whole organizations or independent business units within 55 software companies completed and returned the questionnaire, which represents an effective response rate of 77.9 percent. This is well within the norm of 60 +/- 20 percent for representatives of organizations and mid-level managers suggested for academic studies by Baruch (1999), and considerably higher than prior mail surveys conducted in the Norwegian software and IT industry. A detailed description of the sample was provided in *Section 6.3* and *Table 6.3*.

In the next two sections, we describe the reliability analysis performed to refine the measurement items of the facilitating factors of SPI success.

7.7 Reliability of Measurement Scales

Reliability refers to the consistency and stability of a score from a measurement scale (Anastasi and Urbina, 1997). Since all types of reliability are concerned with the degree of consistency or agreement between two independently derived sets of scores, they can all be expressed in terms of a *correlation coefficient*. Essentially, the correlation coefficient (*r*) indicates the degree of correspondence, or relationship, between two sets of scores. The absolute value of the correlation coefficient can range from 0 to 1.0, with 1.0 perfectly reliable and 0 perfectly unreliable (Anastasi and Urbina, 1997).

The types of reliability computed in practice are relatively few. They are summarized in *Table 7.3* in terms of the different methods for measuring reliability, seen in relation to the number of scale forms required and to the number of scale administrations required. Except for the internal consistency method, the other methods in *Table 7.3* have major limitations for field studies like the one presented in this paper.

The test-retest method requires two independent administrations of the same measurement scales on the same group of people. The alternate form method requires the administration of two equivalent scales to the same individuals, with or without a time interval. The split-halves method works well in field studies because it requires only a single administration of a single form of a scale. However, its problem is how to split the test to obtain the most nearly equivalent halves, because the estimate of the reliability coefficient totally depends on how the items are split.

Table 7.3: Classification of reliability estimation methods (adapted from Anastasi and Urbina, 1997).

	Scale Forms Required	
Administrations Required	**One**	**Two**
One	Split-Halves Internal Consistency (Coefficient Alpha)	Alternate-Form (immediate)
Two	Test-Retest	Alternate-Form (delayed)

An internal consistency technique that overcomes the shortcomings of the split-half method is known as *coefficient alpha* (Cronbach, 1951). This technique computes the mean reliability coefficient estimates for all possible ways of splitting a set of items in two. Hence, coefficient alpha expresses the degree to which items in a scale are homogeneous. The formula for coefficient alpha is given as follows (Cronbach, 1951):

$$\alpha = \left(\frac{n}{n-1}\right)\frac{SD_y^2 - \sum(SD_i^2)}{SD_y^2} \qquad (7.1)$$

In this formula, α is the reliability coefficient for the whole scale, n is the number of items in the scale, SD_y is the standard deviation of total scores on the scale, and SD_i is the standard deviation of individual item scores. Coefficient alpha varies between 0 and 1.0. If there is no true score, but only error in the items, then the sum of variances of the individual items will be the same as the variance of the sum and, consequently, alpha will be equal to zero. If, on the other hand, all items are perfectly reliable and measure the same concept, then coefficient alpha will be equal to one. Coefficient alpha is a conservative estimate that can be regarded as a *lower bound* of reliability (Novick and Lewis, 1967; Carmines and Zeller, 1979).

An internal consistency analysis was performed for each of the six key facilitating factors of SPI success using the SPSS reliability program (SPSS, 1999a, 1999b). We analyzed the correlation of items within each scale (item-item), the corrected item-to-total (item-scale) correlations, the effects of reliability if the item was deleted, and the item standard deviation scores to determine which items were candidates for deletion from the scale. Candidate items for deletion were items with low item-item and item-scale correlations, which would raise alpha if deleted. Other candidate items were items with low variance, which would have low explanatory power. However, before any item was deleted, we made a check to ensure that the domain coverage of the construct would not suffer.

This analysis revealed that to obtain satisfactory values for coefficient alpha while retaining the domain coverage only required one item to be eliminated from the exploitation scale (marked with an asterisk in *Table 7.1*). *Table 7.4* reports the original sets of measurement items associated with the key factors, the items dropped from the original sets to increase alpha, and the reliability coefficients for the resulting scales.

Table 7.4: Reliability analysis results for the key factors of SPI success.

Key Factors of SPI Success	Item Numbers	Number of Items	Items Deleted	Alpha
Business Orientation	1-5	5	None	.81
Leadership Involvement	6-10	5	None	.87
Employee Participation	11-17	7	None	.80
Concern for Measurement	18-23	6	None	.81
Exploitation of existing knowledge	24-29	6	No. 29	.78
Exploration of new knowledge	30-37	8	None	.85

Table 7.4 shows that the reliability coefficients ranged from 0.78 to 0.87, indicating that some scales are more reliable than others are. A satisfactory level of reliability will depend on how a measure is being used. However, in the early stages of instrument research, reliabilities of 0.7 or higher are considered sufficient for narrow constructs (Cronbach, 1951; Nunnally and Bernstein, 1994), and 0.55 to 0.70 for moderately broad constructs (van de Ven and Ferry, 1980). Furthermore, reliability coefficients of 0.50 or above are considered acceptable to compare groups (Fink and Kosecoff, 1998). Besides, Nunnally and Bernstein (1994) argued that increasing reliabilities beyond 0.80 in basic research is often wasteful. Based on these recommendations, our target level of minimum reliability was set in the 0.70 to 0.80 range. Hence, all scales developed for the facilitating factors in this study were judged to be reliable.

In the next section, we describe the detailed item analysis performed to evaluate the appropriateness of each item in each scale.

7.8 *Detailed Item Analysis*

We made a detailed item analysis based on Nunnally's method to evaluate the assignment of items to scales (Nunnally, 1978; Nunnally and Bernstein, 1994). The method considers the correlation of each item with each scale, in order to determine whether an item belongs to the scale as assigned, whether it belongs to some other scale, or whether it should be eliminated. Compared to items with relatively low correlation with total scores, those that have higher correlation with total scores have more variance relating to the common factor among items, and they add more to instrument reliability (Nunnally, 1978). Hence, an item that does not correlate highly with any of the scales, should be eliminated.

There are two types of item-scale correlation: corrected and uncorrected. The *corrected item-scale correlation* correlates the item being evaluated with all the scale items, excluding itself, while the *uncorrected item-scale correlation* correlates the item in question with the entire set of candidate items, including itself. Although the uncorrected item-scale correlation makes good conceptual sense, the item's inclusion in the scale can inflate the correlation coefficient, thus making it spuriously high. The fewer the number of items in the scale, the bigger the difference that the inclusion or exclusion of the item under examination will make. Therefore, we followed the general advice of examining the corrected item-scale correlations rather than the uncorrected (DeVellis, 1991; Nunnally and Bernstein, 1994).

After eliminating the items in the previous steps, the remaining items were correlated with the total scores of each scale, excluding itself. The corrected item-scale correlation matrix for the facilitating factors of SPI success in *Table 7.6* shows that all items have high correlations with the scales to which they were originally assigned. Furthermore, all correlations are above the cutoff of 0.3 recommended by Nunnally and Bernstein (1994).

With the exception of the "feedback" item (item no. 23), all item-scale correlations are substantially higher than the corresponding correlations with all other scales. However, based on the literature review, the expert review, and our own experience, we decided to keep the feedback item within the measurement scale as originally assigned. There are two reasons for this. First, feedback is generally considered as being of utmost importance in all SPI work, and second, we regard feedback as conceptually closer to "Concern for Measurement" rather than to "Exploitation of Existing Knowledge". Therefore, we concluded that all items had been assigned to the appropriate scale, and that no additional items should be deleted.

7.9 Validity of Measurement Scales

For a scale to be valid, it must also be reliable. Validity is differentiated from reliability in that the former relates to accuracy, while the latter relates to consistency. A measurement scale is valid if it does what it is supposed to do and measures what it is supposed to measure (Cronbach, 1971; Anastasi and Urbina, 1997). If a scale is not valid, it is of little use because it is not measuring or doing what it is supposed to be doing. Three kinds of validity are of special concern for this study. These validity concerns are outlined in *Table 7.5* and discussed next.

7.9.1 Content validity

Content validity has to do with the degree to which the scale items represent the domain of the concept under study. Essentially, procedures for content validation involve the systematic examination of the instrument content to determine if it covers a representative sample of the behavior domain to be measured (Davis, 1996). Content validity is built into a test from the outset through the choice of appropriate items (Anastasi and Urbina, 1997).

Table 7.5: Three basic types of validity in measurement instruments.

Types of Validity	**Definitions**
Content validity	The degree to which the items in the measurement instrument represent the domain or universe of the processes under study.
Construct validity	The degree to which the measurement instrument represents and acts like the processes being measured.
Criterion validity	The degree to which the measurement instrument is able to predict a variable that is designated a criterion.

Table 7.6: Corrected item-scale correlation matrix.

	Item			Scale			
Key Factors of SPI Success	**No.**	**1**	**2**	**3**	**4**	**5**	**6**
Business Orientation	1	*.61*	.52	.47	.40	.33	-.03
(Scale 1)	2	*.67*	.43	.37	.36	.47	.18
	3	*.49*	.40	.39	.29	.47	.18
	4	*.69*	.53	.24	.28	.51	.21
	5	*.56*	.51	.46	.40	.38	.15
Leadership Involvement	6	.45	*.67*	.24	.33	.42	.09
(Scale 2)	7	.54	*.81*	.37	.41	.39	.05
	8	.49	*.69*	.30	.33	.34	.13
	9	.55	*.70*	.29	.35	.45	.18
	10	.55	*.65*	.30	.25	.30	.00
Employee Participation	11	.09	.18	*.47*	-.01	.20	.35
(Scale 3)	12	.15	.18	*.55*	.10	.28	.41
	13	.28	.28	*.58*	.02	.39	.31
	14	.09	.12	*.47*	.07	.17	.30
	15	.52	.40	*.54*	.39	.45	.25
	16	.48	.34	*.55*	.34	.44	.32
	17	.16	.19	*.53*	.15	.24	.22
Concern for Measurement	18	.32	.32	.02	*.40*	.13	-.06
(Scale 4)	19	.33	.26	.06	*.61*	.25	-.06
	20	.33	.28	.16	*.72*	.36	.20
	21	.33	.33	.17	*.77*	.29	.18
	22	.43	.36	.34	*.61*	.39	.13
	23	.19	.19	.26	*.31*	.31	.16
Exploitation of existing knowledge	24	.34	.28	.35	.16	*.55*	.20
(Scale 5)	25	.52	.38	.43	.34	*.69*	.15
	26	.32	.33	.22	.28	*.52*	-.05
	27	.45	.32	.28	.51	*.54*	.10
	28	.44	.39	.41	.20	*.47*	.19
Exploration of new knowledge	30	.26	.26	.27	.09	.20	*.59*
(Scale 6)	31	.13	.08	.41	.14	.17	*.67*
	32	.15	.09	.30	.08	.16	*.66*
	33	.10	.05	.36	-.03	.13	*.61*
	34	.14	.14	.44	.30	.20	*.53*
	35	-.07	-.16	.24	-.06	-.13	*.57*
	36	.02	-.03	.15	-.07	-.11	*.50*
	37	.25	.20	.38	.28	.26	*.54*

Our procedure for instrument development followed the general recommendations of Cronbach (1971) and Straub (1989) and included:

(1) An exhaustive search of the literature for all possible items to be included in the scales.

(2) An exploratory study in representative companies to find possible items and scales.

(3) Review of the proposed scales by experts of both psychological testing and SPI. There were also asked for suggestions as to any additions or deletions to the scales.

(4) Pilot test of the scales on a set of respondents similar to the target population. These respondents were also encouraged to include suggestions and criticisms to the contents and wording of the scales.

Hence, we argue that our six measurement scales representing the facilitating factors of SPI success developed in this study have content validity since selection of measurement items was based on generally accepted procedures to obtain content validation.

7.9.2 Construct validity

Construct validity is an operational concept that examines whether the measurement scales represent and act like the attributes being measured (Cronbach, 1971; Nunnally and Bernstein, 1994). Assuming that the total score of a scale is valid, the extent to which an individual item measures the same thing as the total score is an indicator of the validity of that item. Furthermore, factor analysis is considered to be "a powerful and indispensable method of construct validation" (Kerlinger, 1986). Thus, in addition to item-total correlations, the construct validity of the scales in the measurement instrument was assessed using factor analysis.

Although many investigators advocate the construction of measurement instruments through factor analysis, Spector (1992) and Nunnally and Bernstein (1994) argued that factor analysis should rather be used *after* the instrument is constructed. The main reason for this is that the construction of a measurement instrument should be guided by theories, rather than by random efforts to relate things to one another through "shotgun statistics". This fits the approach in this research where the constructs of interest are based on a substantial body of prior research and have been explicated prior to any item development.

The construct validity of the six measurement scales was evaluated by analyzing the items of each of the scales using principal components analysis. However, this analysis, as all factor-analysis procedures, includes subjective judgement when it comes to determining the number of factors and their interpretation (Spector, 1992).

Several guidelines have been developed to assist in deciding how many factors to extract. The two most widely used criteria are Kaiser's (1960, 1970) *eigenvalue rule* and Cattell's (1966) *scree test*. The eigenvalue rule is based on retaining only factors that explain more variance than the average amount explained by one of the original items (i.e. components with eigenvalue > 1). However, when Zwick and Velicer (1986) examined the effect of different criteria for extracting factors, they found that Kaiser's method tended to severely overestimate the number of components. Furthermore, Cliff (1988) called the rational behind the eigenvalue rule into question, and despite its wide use, Nunnally and Bernstein (1994) did not recommend it because it tends to suggest too many factors.

The scree test is based on a subjective examination of the plot of eigenvalues for each successive factor, looking for an "elbow" in the plot. Cattell's guidelines call for retaining those factors above the "elbow" and rejecting those below it.

Each item also has a final loading on each factor. These loadings are the correlations between items and the factors, thus the square of a loading can be seen as the amount of variance common to both. Comrey (1973) suggested that loadings in excess of 0.45 could be considered fair, those greater than 0.55 as good, those of 0.63 very good, and those of 0.71 as excellent. For a study with a relatively small number of cases, the quality of a factor must be assessed both in terms of the number and size of its loadings. Stevens (1992) suggested that a reliable factor must have four or more loadings of at least 0.6 when the number of cases is below 150.

We used a combination of eigenvalues, cut-off points of the scree plots, and factor loadings as a guide for interpreting the dimensionality of the scales. Furthermore, each scale was assumed to be a separate construct. *Table 7.7* shows the eigenvalues and item loading ranges for each scale. Analysis of the eigenvalues showed that five of the six scales formed a single factor. In the case of the exploration scale, two components seemed to emerge with eigenvalues greater than 1.0. However, the eigenvalue of the second factor was only slightly above this threshold (1.02).

Next, we examined both the scree plot and the item loadings in order to further interpret the dimensionality of the exploration scale. The scree plot showed a clear break after the first component (see *Figure 7.2*). Furthermore, all item loadings for this component were greater than 0.6. The combined result of these analyses indicates fairly strong support for the hypothesis that the exploration scale can be considered as one construct.

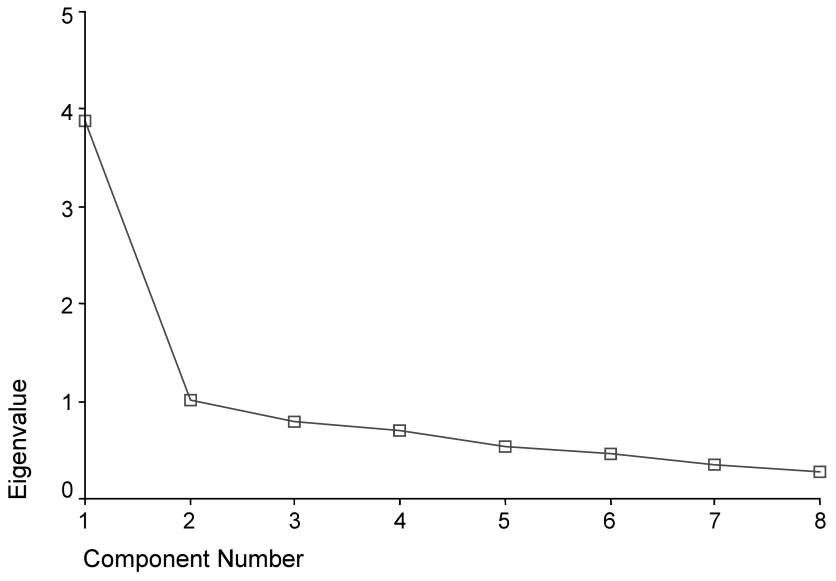


Figure 7.2: Scree plot for the Exploration scale.

Table 7.7: Summary of factor matrices for each construct.

Scale	Eigenvalue	Item Loading Range for Component 1	# Items with Loadings > 0.6
1. Business orientation	2.9	.72 to .82	5 (out of 5)
2. Leadership involvement	3.3	.78 to .89	5 (out of 5)
3. Participation	3.2	.62 to .72	7 (out of 7)
4. Measurement	3.1	.43 to .83	4 (out of 6)
5. Exploitation	2.7	.65 to .84	5 (out of 5)
6. Exploration	3.9	.61 to .78	8 (out of 8)

In brief, the results of the factor analysis support the view that each of the measurement scales in the questionnaire has a high degree of unidimensionality. Hence, there is tentative evidence of construct validity for all six scales.

7.9.3 Criterion validity

Criterion-related validity, sometimes called external validity, is concerned with the degree to which the scales under study are related to an independent measure of the relevant criterion (Anastasi and Urbina, 1997).

Organizational performance is, as we argued in *Section 5.4*, the ultimate criterion for any SPI activity. The six facilitating factors, thus, have criterion-related validity if, taken collectively, they are highly and positively correlated with performance in a software organization. Performance is a complex construct, however, reflecting the criteria and standards used by decision-makers to assess the functioning of a software organization. That is, performance is a value judgement on the results desired from an organization (van de Ven and Ferry, 1980).

Objective measures of performance are problematic in the IS field (Henderson and Lee, 1992). Objective productivity measures – such as lines of code per person per month – are often unavailable, are subject to manipulation, and may reflect the specific accounting practices of a site rather than "actual" performance. Further, using objective measures assumes comparability across software projects, does not control for differences between projects or unique situational constraints, and thus raises a new set of methodological and measurement issues.

Investments in SPI share many features with research and development investments, frequently having corporate-wide, intangible, and long lasting effects. Therefore, quantitative measures and financial estimates tend to be difficult to obtain and easy to manipulate. The main distinction between the objectivity of different success measures is not a matter of using quantitative or financial figures as opposed to developers' or managers' perceptions. For example, the return on investment, net present value and payback periods are often regarded as objective measures. However, because of the many difficulties in predicting and assessing costs, and especially benefits, such investment analyses are usually based on experts' subjective judgement (Saarinen 1996; Zahran, 1998). In many cases, the subjective measures may be better than the financial measures (Saarinen, 1996). This is mainly due to complicated contingencies that are best judged by those who are responsible for the object of evaluation.

Furthermore, value judgements on desirable performance outcomes often change over time in a way that threatens the applied relevance of longitudinal assessments of organizations. Finally, decision makers often disagree on a given set of criteria to evaluate organizational performance. However, performance measurement does not require a consensus on effectiveness goals, criteria, and standards. An organizational assessment simply requires that the definitions of performance be made explicit and that the researcher determines whose value judgements and criteria will be operationalized and measured (van de Ven and Ferry, 1980).

Based on the preceding discussion, and that subjective performance measures are widely used and accepted in organizational research (Lawrence and Lorsch, 1986; Powell, 1995), we considered a subjective evaluation approach with multi-item constructs as an appropriate way to assess organizational performance. Furthermore, previous research has shown that subjective assessments of performance provided by knowledgeable managers have a high level of convergence with other objective measures of performance (Venkatraman and Ramanujam, 1987).

Generally, the intent of SPI is increased competitive advantage through improved software quality, increased productivity, and decreased lead-time for product development. In other words, SPI should lead to "better, faster, [and] cheaper software development" (Sanders, 1998). Based on these general recommendations, we operationalized and measured SPI success along the two dimensions defined in the dynamic model of SPI developed *in Chapter 5*, based on two multi-item measures. Thus, each manager was asked to rate, on 5-point bipolar Likert scales, (1) the level of perceived SPI success and (2) the performance of their organization for the past three years with respect to cost reduction, cycle time reduction, and customer satisfaction. This is similar to the approach used by Teng *et al.* (1998) to measure reengineering project success and the relationship between characteristics of reengineering projects and implementation success.

Two items were used to measure the level of perceived SPI success, while three items were used to measure the level of organizational performance. As for the facilitating factors, all items were written specifically for this study. These items are included in the validated measurement instrument in *Appendix E*. The ratings for the two performance dimensions were averaged to form a single measure of overall SPI success. The reliability coefficient for the five items of the SPI success measure was 0.76.

The criterion validity of the measurement instrument was found by computing the multiple correlation (R) between SPI success and the measures of the six facilitating factors. The multiple correlation coefficient was 0.76 and the adjusted R-square was 0.56, thus, explaining 56 percent of the variation in SPI success. Thus, taken together, the six factors have a high degree of criterion-related validity. In the next chapter, we examine the joint contribution of the independent variables to the explanation of SPI success in more detail.

7.10 Chapter Summary

In this chapter, we have described an extensive investigation to the development of an instrument for measuring the key factors of success in SPI. The major premise was that it is critical to understand the important factors affecting SPI success in order to improve software processes. However, central to this understanding is the development of an instrument for measuring these factors.

The results of reliability and validity analyses showed that the instrument has desirable psychometric properties. This is good for an instrument that is composed entirely of new scales, particularly since the scales do not contain large numbers of items. However, demonstrating validity is a continuous process, and validity is a matter of degree rather than an all-or-none property. Moreover, one validates the *use* to which a measurement instrument is put rather than the instrument itself (Nunnally and Bernstein, 1994). Consequently, it is only through the accumulation of data from multiple studies of instrument usage that we can make strong claims of validity.

Hence, as the basis for testing the hypotheses developed in *Chapter 5*, we have an empirically derived *model* of SPI success, as well as a validated and reliable *instrument* to measure the key factors of success in SPI. We are, thus, not only able to identify the key factors for success in SPI, but also to measure the strength of their relationships with SPI success as well as their joint contribution and relative importance to the explanation of SPI success.

In the next chapter, we will use this instrument to test the hypotheses and explore the relationships that were proposed in *Chapter 5*.

CHAPTER 8
Empirical Results

"*Whoever desires constant success must change his conduct with the times*"

– Niccolò Machiavelli

In the preceding chapter, we described the development of an instrument, with satisfactory psychometric properties, for measuring the key factors of success in SPI.

In this chapter, we report on the results of the main study of using this instrument to test the hypotheses regarding the key factors for success in SPI and explore the relationships between the success factors and the contextual variables. First, however, we report on the three supplementary case studies regarding (A) the experience with goal-oriented measurement, (B) the experience with participative software process assessment, and (C) the utility of formal routines to transfer knowledge and experience.

8.1 Case Study A: Experience with Goal-Oriented Measurement[1]

In this section, we describe our experiences on the application of goal-oriented measurement using **ami** and GQM as the basis for systematic quality improvements in telecommunications software development within Nera AS (ESSI project 21780: TELMET).

8.1.1 Defining GQM goals

The improvement experiment started with the decomposition of primary goals from the company's business strategy into sub-goals and subsequently into GQM goals. Four goals were formulated according to the GQM template and ranked according to their importance. This led to the identification of a set of more than 90 primitive metrics. It soon became clear that this would lead to large costs for data collection and analysis. In addition, it could be a considerable extra load on the personnel in the baseline project.

[1] The results reported in this section are based on (Dybå *et al.*, 1997; Stålhane *et al.*, 1997).

As a result of these considerations, it was decided to reduce the number of goals from four to two. This reduced the number of primitive metrics from 90 to 40, which was considered to be a manageable number both for the project and for the people involved in the data analysis and other follow-up activities. The selected goals were:

- **Reliability**:
  Analyze the development process *for the purpose of* improvement *with respect to* causes of reliability *from the viewpoint of* the software development team *in the following environment*: Nera AS, Tele-Scada.
- **Verification and validation**:
  Analyze the development process *for the purpose of* improvement *with respect to* V&V effectiveness *from the viewpoint of* the software development team *in the following environment*: Nera AS, Tele-Scada.

As soon as the GQM goals were defined, we stared to develop the measurement plan.

8.1.2 Defining the measurement plan

The work on the measurement plan started with a workshop on applying the GQM process, followed by a hands-on exercise. The workshop was held with the people representing the viewpoints of the selected GQM goals. The GQM abstraction sheet was introduced to the participants at the beginning of this workshop, and the participants were divided into groups who filled out one abstraction sheet per goal per group.

The resulting abstraction sheets were combined, and conflicts and inconsistencies were resolved by gathering all involved people in a joint session. Each group presented their viewpoints to the rest of the team, and the metrics promoter was responsible for follow-up interviews, final clarifications, and for finalizing the abstraction sheets - one sheet per goal. *Figure 8.1* shows part of the final GQM structure for Nera Tele-Scada AS, while *Table 8.1* shows the resulting GQM abstraction sheet for the goal “reliability”.

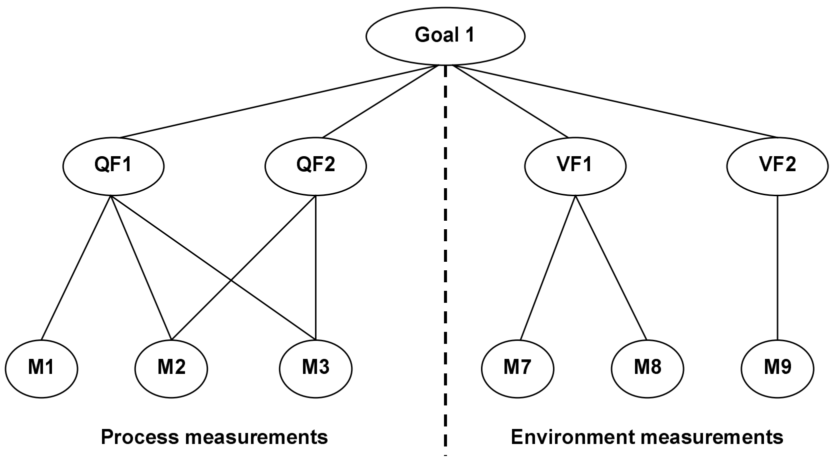


Figure 8.1: Part of Nera’s GQM tree.

Table 8.1: GQM abstraction sheet for goal reliability.

Goal	Object	Purpose	Quality focus	Viewpoint	Environment
G1	Development process	Better understanding	Causes of reliability	Sw development team	Nera AS, Tele-Scada

Quality Focus	Variation Factors
1 What is the incident density during system test?	1 How many hours are used for review preparations?
2 What is the fault density during system test?	2 How good are the requirements traceability?
3 What is the fault density after release?	3 How experienced are the project members?
4 How are faults distributed over severeness categories?	4 How much application experience do the project members have?
5 How are faults distributed over phase of introduction?	5 What is the resource situation for the project?
6 How are faults distributed over fault types?	6 How much did each project member participate in the OSD phase?
7 How many hours are used for correcting a fault after release?	7 How much did the PM participate in the requirements phase?
	8 How many large changes of technology was there in the project?
	9 How well was the quality controlled in the project?

Baseline Hypothesis	Impact on Baseline Hypothesis
1 M1/M2 = X% (+/- Y%)	1 An increase in VF1 will reduce QF1, QF2, QF3
2 M3/M2 = X% (+/- Y%)	2 A low value for VF2 will increase QF1, QF2, QF3
3 M4/M24 = X% (+/- Y%)	3 An increase in VF3 will reduce QF1, QF2, QF3
4 X% Fatal, Y% Critical, Z% Non-critical	4 An increase in VF4 will reduce QF1, QF2, QF3
5 REQ = X%, OSD=Y%, DES=Z%, DET = S%, CODE = T%, INT = U%, SYS=V%	5 A bad VF5 will increase QF1, QF2, QF3
6 Init = X%, Logic=Y%, Data=Z%, Opt=S%, Scaling=T%, IntCom=U%, Use = V%, Env=W%	6 An increase in VF6 will reduce QF1, QF2, QF3, QF7
7 X hours/fault	7 An increase in VF7 will decrease QF1, QF2, QF3
	8 A large VF8 will increase QF1, QF2, QF3, QF7
	9 A large VF9 will decrease QF1, QF2, QF3, QF7

It turned out that the process of doing the initial guesstimates of the metrics' values (the "baseline hypothesis" part of the work sheet) also was an opportunity to collect additional knowledge and refine our GQM tree. Insisting on these initial guestimates caused the developers to reconsider some of the suggested metrics. It is one thing to agree that something can be measured – to find plausible values is something quite different. When we for instance started to provide guestimates for the number of remarks per page, the developers were quick to point out that this must differ strongly from code to documents. Thus, the metric "number of remarks per page" was split into "number of remarks per document page" and "number of remarks per code page" before the developers were able to supply reasonable estimates for the values. The developers later characterized this part of the work as a difficult but useful learning process.

We made sure that the GQM models and the measurement plan only contained primitive metrics. This way, the answer to each question in the GQM models was a simple calculation, where the measurements were combined into quantified items that directly reflected the attributes (question) whose value (answer) we were interested in. Subsequently, the use of standard graphs such as scatter plots, pie charts, and histograms visualized these answers.

Based on the metric definitions, we described how each metric should be measured, when it should be measured, and by whom. Together, this made up the measurement plan for the baseline project.

Table 8.2: Metrics definition template.

Item	Description
Id	Unique identifier of the primitive metric.
Name	Name of the primitive metric.
Definition	Unambiguous description of the metric in terms of the project's environment.
Collection procedure	Description of the collection procedure.
Expected value	The expected value, taken from the "hypothesis" part of the GQM abstraction sheet.
Responsible	Who is responsible for collecting the data.

As soon as we had a small data set – 10 values – we performed some tentative analyses. Besides getting a start on the results of the process improvement experiment, this early analysis had several other important purposes. It helped us to check that:

- The measurement plan was precise enough for the persons responsible for the data collection.
- We had all the data needed to obtain useful results from the analyses.
- The chosen data presentation format was suited for the feedback meetings.
- The data selected would provide the developers with sufficient information for process understanding and later improvements.

The final measurement plan consisted of a set of tables – one per metric – and a presentation description for the answer to each question (see Pulford *et al.*, 1996 for additional information). In addition, we made a summary table that contained the most important information for all the metrics; when the metrics were to be collected and references to the relevant registration forms. The template for the metrics definitions is shown in *Table 8.2*.

8.1.3 Data collection and validation

The strong focus on data definition, data collection, and participation at the start of the GQM process created ownership and interest among the developers. The most important result from the GQM process was that the developers and management felt that the data were important for their job. However, the company's commitment to continuous process improvement was probably of equal importance, especially to the managers.

Four types of data collection was going on during the GQM process; data was collected:

- While working on the GQM work sheets. This was the developers' prior beliefs for the answers to the questions.
- While executing the measurement plan. This was the actual data collected as product and process metrics.

- During the feedback sessions. This was knowledge pertaining to relations between parameters and between parameters and the project's environment.
- During data interpretation and data analysis. This was knowledge pertaining to suggestions for data refinement; which characteristics are relevant for a further refinement of the data.

When we considered each metric in isolation, we could make do with only the two first types of data collection. However, when we started to analyze the data in order to understand a part of the process, the other types showed their importance.

While we came a long way in planning, collecting, and formalizing the two first types of data, only small attempts were made for the third and fourth types. One of the reasons why this was difficult was that much of this data collection depended on knowledge that could only partially be documented.

As mentioned in the previous subsection, we validated the data as soon as they became available. This led to a better understanding of the data that we were collecting. No new metrics were introduced during this work, but some of them needed a redefinition.

The metric "Number of review remarks" is a case in point. When this metric was checked for correlation with the document complexity, as assessed by the developers, it became clear that the variations were not as expected. The reason was clear: Since the number of remarks included the whole spectrum of mistakes – from typographical errors to deep logical errors – the contribution from the small problems, roughly proportional to the size of the document, was much larger than the contribution from the document's complexity. The developers considered the logical errors to be much more serious and thus more important to understand. As a result of this, it was decided that we should only collect the number of remarks concerning logical errors and disregard the rest of the review remarks.

8.1.4 Data analysis

As with the GQM, the data analysis also started with a short, introductory course. The course consisted of a half-day introduction to statistical data analysis, followed by half a day of exercises. The main parts of this course were plotting techniques, *t*-statistics and ANOVA. In order to make the exercises relevant to the participants we used data collected from the department's development projects.

The general approach to data analysis was to plot the data before any statistical analysis was attempted. Also, in order to make data collection, registration, and analysis simple, we decided to use the MS Excel spreadsheet with its built-in statistical package, which turned out to be sufficient for our needs. Thus, the analysis strategy was as follows:

(1) Plot the data sets under consideration. Two things could happen: Either a clear picture would emerge or it did not.

(2) If the picture emerging from the data plot was clear, a regression analysis or ANOVA was attempted in order to see how much of the data variation that could be explained by a simple model.

If, on the other hand, no clear picture emerged, we would try some standard data transformations like the *log* or the *logit* transformation. If this did not help, we would take this as an indication that we needed to look for other explanatory parameters in order to understand this particular relationship.

(3) In all cases, the need for interpretation and suggestion for additional parameters were discussed together with the developers in the feedback sessions. This could lead to the need for new metrics, new definitions for some of the current metrics, or a new procedure for data collection.

In order to present several metrics or questions together, we used the Kiviat diagram – also known as the radar chart. This presentation method was especially useful for comparing a set of measurements against some goal or a set of ideal results.

The most significant lesson from the data analysis, however, was the great importance of expert opinion. These opinions were put forward and discussed together with the possible interpretations of the data in dedicated learning forums – the feedback sessions.

8.1.5 Feedback sessions

Feedback sessions play an important role in the GQM method. It soon turned out that these sessions were the most important part of measurement-based understanding and improvement in the TELMET project. These sessions were important for several reasons. They helped in:

- Interpreting the data. The developers' insight into the development process gave important input for the interpretation of the measurements.
- Suggesting additional parameters that could help to make the collected data "make sense". This could give rise to new metrics that needed to be collected.
- Moving from measurement and process understanding toward process improvement. This was achieved by combing the developers' current experience and knowledge with the new insights provided by the measurements and the analysis.

All these effects were observed in the project and resulted in an important learning process. Thus, knowledge and experience are of paramount importance for the data definition, data analysis, and interpretation.

8.1.6 Lessons learned

The most important lesson learned was that the feedback sessions and close interaction between the project team and the metrics team, are the most important mechanisms both for the overall success of the measurement program and for data analysis and interpretation. Besides, measurement-based improvement was identified as an important business concern with strong management support. This was undoubtedly important for the success of the process improvement experiment.

The use of the GQM abstraction sheet was very useful, although the language used could be both hard to understand (e.g. quality focus) and confusing (e.g. impact on baseline hypothesis) in the beginning. After a short initial phase, however, the team members became familiar with its use.

The needed effort to formulate precise questions in order to use only primitive metrics paid off, as we were able to simplify both data presentations and analysis related to the questions asked. However, we started to plan the experiment with four goals and their corresponding questions and metrics. This was too ambitious, and we had to reduce this to two goals. The number of questions and metrics was not changed, but during data analysis we found that some were more important than others were, and that we could have limited ourselves to concentrate on these only.

The following is a list of some of the traps we have fallen into or barely avoided during our application of GQM in the TELMET project:

- GQM is much more than a method for defining a model for software metrics collection. It is first and foremost a method for the developers to externalize their knowledge and experience. From this point of view, each step in the GQM method is an opportunity to learn more about the development process or to externalize more of the large amount of experience and knowledge already available in the organization.
- GQM is *not* a method for data analysis. It *is*, however, a method for improving and closing the loop of defining, collecting, analyzing, and interpreting product and process data.
- The term “Hypothesis” used in the GQM work sheet is highly misleading. There are *no* hypotheses in the statistical sense in the GQM process. It is always a question of prior belief, as this term is used for instance in Bayesian statistics.
- The GQM process is *not* about quality. The process can be used in any measurement setting where a large part of the knowledge needed for improvement only is available as human knowledge and experience. The frequent use of the term “quality” in the GQM work sheet can be considered an attempt to surf on the quality wave. On the other hand, the GQM process *is* an important tool for data collection in process improvement.

Also, the GQM method as it is presented now does not give any guidance when it comes to formulating the questions once the goals are specified. One possible way to do this could be to proceed as follows:

1. Specify the goals.
2. Formulate hypotheses related to each goal.
3. Use these hypotheses to formulate the needed questions.

E.g. if the goal is to understand the cause of X, the hypothesis could be formulated as “A will improve X”. From this hypothesis we could derive questions like “What is the value of A?”, “How will Y influence A – or X?” and so on. This would give a more focused, and thus more efficient, way of deriving questions in the GQM tree.

8.2 *Case Study B: Experience with Participative Software Process Assessment*[2]

In this section, we describe our experience with the adoption of a general approach for organizational assessment and its specialization to the domain of software development. The process involved researchers and practitioner acting together in a participative approach to diagnose problems in software development in four organizations using the basic principles of survey feedback (see e.g. Baumgartel, 1959; Neff, 1966), which is a specialized form of action research.

8.2.1 Assessment method

The assessment process is an adaptation of the evaluation model developed by van De Ven and Ferry (1980) and consists of six steps, as shown in *Figure 8.2*, and described next.

In the first step, *assessment initiation*, the insiders and outsiders of the organization should clarify their respective roles and the objectives of the assessment by answering the following questions:

(1) What are the purposes of performing software process assessment?

(2) Who are the users of the assessment, and how will the results be used?

(3) What is the scope of the assessment in terms of organizational units and issues?

(4) To what extent is there a commitment to using scientific methods (e.g. psychometric principles) to design and implement the assessment?

(5) Who should conduct the assessment, and what resources are available?

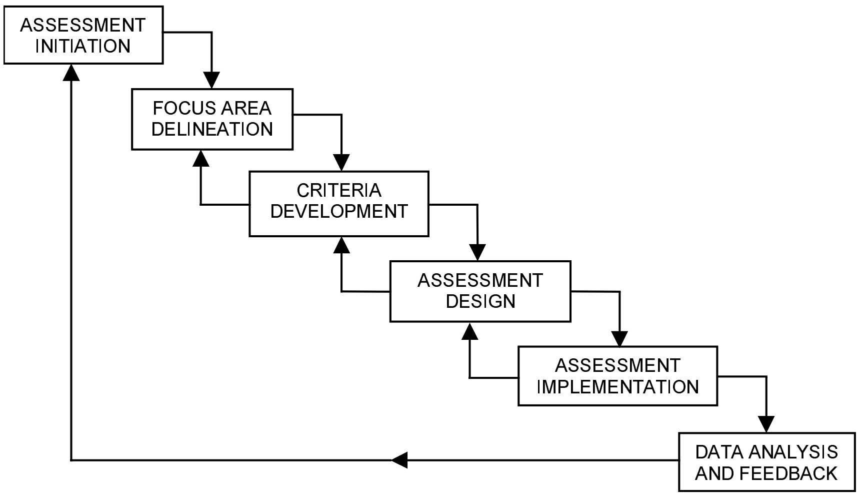


Figure 8.2: Participative assessment process.

[2] The results reported in this section are based on (Dybå and Moe, 1999).

It is important that due considerations are taken in answering these questions, since they are crucial for determining whether an assessment is relevant in the first place, and for tailoring the process and content of the assessment to the specific needs of the organization.

The second step, *focus area delineation*, is an exploration of the overall issues identified for the assessment in step one. In our experience, most companies do not have a shared understanding of their specific goals. A conscious analysis of commonly used high-level performance goals and focus areas in standards and reference models can, therefore, be useful as a starting point for group discussions in this step. Examples of such focus areas are software processes (e.g. customer-supplier, engineering, support, management and organization), competitive priorities (e.g. price, quality, flexibility, and service), organizational learning (learning from past experiences, learning from others, and current SPI practices), and perceived factors of success.

In the third step, *criteria development*, multiple operational criteria are developed for each of the high-level goals. The process of criteria development requires that practitioners select and define concrete characteristics that are to be measured and used as indicators of goal attainment. A decision also has to be taken regarding the use of aggregate or composite measures.

To operationalize the criteria, one question is defined for each characteristic such that the theoretical abstractions can be closely related with everyday work situations. We adopted the format used in the European Software Institute's 1995 Software Excellence Survey, such that two subjective rating scales accompany each question: one to rate the current strength or practice level and one to rate the future importance (see *Figure 8.3*).

Furthermore, to help the companies, we developed a standard questionnaire that could be used as a starting point for internal discussions and for the development of tailor-made questionnaires. The standard questionnaire is based on our experiences of performing process assessments in six companies during the SPIQ pre-project phase, and includes four sections. The first section, on competitive priorities, was adopted from the aforementioned ESI survey. The second section was adopted from the software process areas in ISO/IEC 15504. The third section concerned SPI processes and learning from past experiences and the experiences of others. Finally, the fourth section was concerned with finding the most important factors enabling SPI success in the organization (see *Section 7.2*).

The issues pertinent to step four, *assessment design*, relate to where the assessment will be conducted (organizational units), the role of the insiders and outsiders, the time horizon of the assessment, the unit of analysis, as well as deciding what the sample would be, how the data will be collected, how aggregate concepts will be measured, and how the data will be analyzed.

Current strength						**Future Importance**				
1	**2**	**3**	**4**	**5**	**DELIVERY**	**1**	**2**	**3**	**4**	**5**
☐	☐	☐	☐	☐	*Ability to deliver on schedule*	☐	☐	☐	☐	☐

Figure 8.3: Typical question format from the questionnaire.

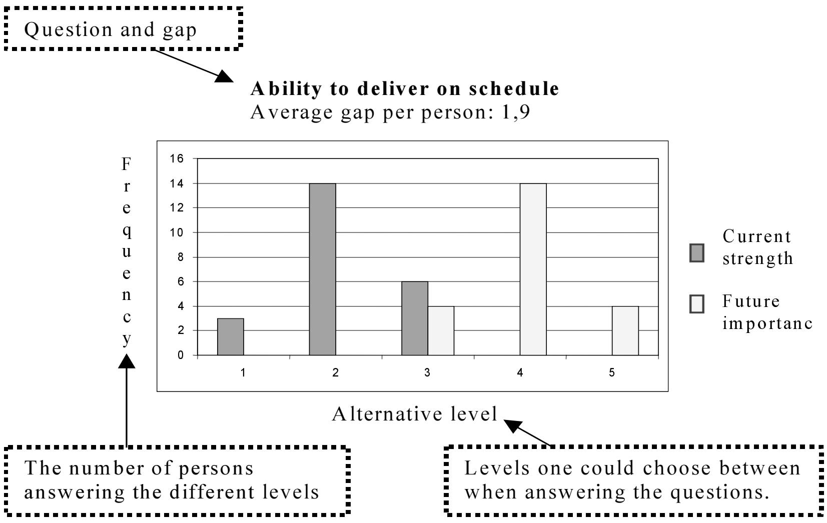


Figure 8.4: Example of an illustration of preliminary findings.

When aggregate or composite measures are used, one should be careful about deciding the corresponding requirements of psychometric properties (see e.g. Nunnally and Bernstein, 1994). That is, unless the composite scales in the questionnaires are constructed and evaluated along the lines associated with psychometric tests, they may produce assessment results that are seriously misleading.

It is important, however, to note that the more rigorous the assessment design becomes, the greater the time, costs, and other resources expended on the assessment are likely to be. Therefore, one should ask the question at every decision point whether the benefits that result from a more sophisticated design to ensure accuracy, confidence, generalizability, and so on, are worth the investment of more resources.

In step five, *assessment implementation*, the assessment is implemented according to the procedure decided upon in the previous step. The main considerations during this step are completeness and honesty in data collection procedures and the recording of unanticipated events that may influence the assessment results.

The major concerns during step six, *data analysis and feedback*, are to provide opportunities for respondents to participate in analyzing, interpreting, and learning from the results of the assessment. And, furthermore, to identify concrete areas for improvement. There are many ways in which this could be done. We have relied upon half-day workshops in which preliminary findings on initial questions and problems are presented verbally, in writing, and with illustrations.

These workshops begin with a review of the objectives of the assessment, the focus areas, and the design and implementation of the assessment. Findings regarding the scores on current strengths and future importance are presented in terms of a gap analysis. Normally, the participants raise a multitude of questions and issues when the findings are presented, and they take part in group discussions and reflections as they review and evaluate the preliminary findings. Some of the questions can be clarified and answered directly with the data at hand, other questions can be answered by reanalyzing the data, and finally some issues are raised which cannot be resolved with the current assessment data. In the last case, a decision has to be taken regarding further data collection.

Typically, we used scatter plots, bar charts, and histograms to illustrate preliminary findings from an assessment in order to highlight gaps between current levels of practices and future importance and the dispersion of responses both within and between groups.

Figure 8.4 shows an example of an illustration that was used in a feedback session presenting preliminary findings from an assessment. Some of the information was presented verbally. The extra information for this graph was "100% of the respondents have a gap that is larger than or equal to one".

8.2.2 Lessons learned

(1) Assessment initiation

- By involving more than one group in the assessment, there exist possibilities for multiple views and interest in the discussion and analysis phase.

(2) Focus area delineation and criteria development

- A team should be put together in the company performing the assessment in order to construct a tailor-made questionnaire.
- The time needed to complete the questionnaire should not exceed 30 minutes (about 60 questions). If there is a problem with not covering all the areas identified during the initiation phase, the company should conduct more than one assessment.
- There should be a close cooperation between the outsiders and the insiders. Companies often have problems distinguish between important and less important questions. In such cases, people from the outside will be able to help by posing "stupid" questions.
- It is very important for the outsiders to get to know the company well enough to be able to give useful advice. To achieve this, it is critical that they work closely together with the people in the company performing the assessment.

(3) Assessment design

- Hold a presentation for the persons participating in the assessment. This presentation should be both motivating for the assessment, and secure that everybody has the same understanding of the questions and the goals of the assessment.
- During the assessment, there will always be discussions regarding the interpretation of questions. It is therefore advantageous to let the participants conduct the assessment at the same time.
- All information should be treated as confidential. If not, there will always be someone not speaking from the heart.

(4) Assessment implementation

- Do not wait too long before performing the feedback-session. This may lead to loss of SPI focus in the department/company.

- If definitions of questions are discussed during implementation, it is wise to document the conclusions of these discussions. During the analysis and feedback session, it is highly possible that these questions will be discussed again, and the participants will not always remember how they interpreted these during the assessment.

(5) Data analysis and feedback

- Do not leave the feedback session without identifying concrete areas for improvement. These areas will be input to the next phase of improvement – action planning.
- To have a useful discussion in this session, make sure the group participating is not too large or too small. If there is a big group, the assessment leader in the company should prepare the data to suggest some key areas before the half-day workshop. The ideal size of the group is 8 – 14 people.

Maybe the most valuable lesson learned was the need for the assessment to be closely aligned with and tailored to the company's overall strategy. Without this it is hard to get acceptance by the managers, leading to fewer resources and low priority. It is also necessary with a tight time-schedule. Waiting too long between the steps will only lower the interest and motivation.

8.3 Case Study C: Exploring the Utility of Formal Routines to Transfer Knowledge and Experience[3]

To further validate the learning cycle in our dynamic model of SPI, we made a separate empirical study with 13 developers and 10 managers in five IT companies on the utility of formal routines to transfer knowledge and experience. Of particular concern was the effect of participation on the relationship between organizational memory and local knowing and potential differences in opinion between developers and managers regarding the utility of the organizations' routines.

Data collection focused on answering the following four research questions:

Q1: What is the knowledge of the routines being used?

Q2: How are these routines being used?

Q3: How are they updated?

Q4: How effective are they as a medium for transfer of knowledge and experience?

The interview guide used to answer these questions is included in *Appendix B*.

8.3.1 Knowledge of routines

All respondents had a fairly good knowledge of the routines that were in place in their respective companies. In fact, two thirds of the respondents showed good knowledge about

[3] The results reported in this section are based on (Conradi and Dybå, 2001).

the content of the routines. *Table 8.3* illustrates this, and shows how well developers and managers were able to describe the specific contents of the routines in their company.

However, when it came to knowledge about how the routines were introduced, 50% of the developers did not know anything about this process. On the other hand, only one manager did not know about the introduction process. All in all, it turned out that about 30% of the developers and 70% of the managers had actively *participated* in the introduction of routines (*Table 8.4*).

It seemed to be a common understanding regarding the *objective* of having formal routines. Most respondents said that such routines were useful with respect to quality assurance. Other respondents said that they would enable a more unified way of working. However they emphasized that:

> "Routines should not be formalistic, but rather useful and necessary".

Respondents in the three ISO-9001 certified companies claimed that their routines were first and foremost established to "get the certificate on the wall", and that the quality of their software processes had gained little or nothing from the ISO certification. One of the respondents expressed his views on this by the following example:

> "You might be ISO certified to produce lifebelts in concrete, as long as you put the exact same amount of concrete in each lifebelt."

Although some of the respondents were critical to the routines, stating that:

> "10% of the routines are useful, while the remaining 90% is nonsense"

Most respondents, nevertheless, had a good impression of the routines, typically stating that:

> "Routines are a prerequisite for internal collaboration."

> "Routines are a reassurance and of great help."

8.3.2 Use of routines

Software developers and managers agreed on the degree to which the routines were used. In general, they answered that about 50% of the routines were in use, and that the more experienced developers used the routines to a lesser extent than the more inexperienced developers do.

Table 8.3: Knowledge of company routines

	Software developers (*n* = 13)		**Managers (*n* = 10)**	
	Frequency	Percent	Frequency	Percent
Little	-	-	-	-
Some	6	46	2	20
Much	7	54	8	80

Table 8.4: Degree of involvement during introduction of routines.

	Degree of involvement			
	Low		**High**	
	Frequency	Percent	Frequency	Percent
Developers	9	69	4	31
Managers	3	30	7	70

Furthermore, it was a common agreement that:

> "There is no point in having routines that are not considered useful".

However, the status of the routines among the software developers was highly divergent, as seen from the following statements:

> "The routines are generally good and useful, but some developers are frustrated regarding their use."
>
> "The system is bureaucratic – it was better before, when we had more freedom to decide for ourselves what should best be done."
>
> "The routines are easy to use."
>
> "Routines are uninteresting and revision meetings are boring."

8.3.3 *Updating of routines*

None of the companies had scheduled revisions as part of the process for updating their routines. Most answers to this issue were rather vague. Some respondents explained that such revisions were informally triggered, while other respondents did not know how to propose and implement changes to existing routines.

However, respondents from all of the companies, both managers and software developers, said that all employees in their respective companies could participate in the revision activities if they wanted to.

8.3.4 *Routines as a medium for transfer of knowledge and experience*

The answers to this issue varied a lot, and indicated highly different attitudes regarding the effectiveness of formal routines for knowledge and experience transfer. Particularly, it seemed to be a clear difference in judgment between software developers and managers. While seven of the ten managers regarded written routines as an efficient medium for knowledge transfer, none of the developers did. Furthermore, half of the developers considered such routines to be inefficient for knowledge transfer, while only one of the managers shared this view.

Typically, managers said that written routines were important as means for replacing the knowledge of the people that had left the company. Software developers, on the other hand, did not make such a clear connection between experience, knowledge transfer, and formal routines. One software developer said that different groups within the company never read each other's reports, while another developer maintained that it would take too much time to learn about the experience of the other groups. Several of the developers explained their views by stating that the documentation was not good enough, it was hard to find, boring to use, and that it takes too much time.

When asked about useful alternatives to written routines, the respondents answered that they regarded some kind of "Experience base" or "Newsgroup" as the highest ranked alternative. Other high-ranked alternatives were "Socialization", "Discussion groups", "Experience reports", "Group meetings", and "On-the-job training". *Table 8.5* shows these alternatives in rank order (1 is best) for software developers and managers respectively.

We also asked the respondents about what they regarded as the most important barriers against transfer of knowledge and experience. Nearly all of them said that such transfer, first and foremost, is a personal matter depending on how much each individual whishes to share their lessons-learned with others. Furthermore, the willingness to share depends on available time, personality, self-interest, and company culture.

As shown in *Table 8.6*, seven (six developers and one manager) of the 23 respondents answered a definite "No" to the question regarding the efficiency of written routines as a medium for transfer of knowledge and experience. Likewise, seven respondents (managers only) answered an equally clear "Yes" to the same question. The last nine respondents answered somewhere in between, and said that in some cases written routines could be effective, while in other situations they would rather be a barrier.

Due to the rather small sample size in this part of the study, and the low expected frequency in several of the cells in *Table 8.6*, we compared the respondents' assessments of the routines and their job function using Fisher's exact probability test. With this test, the exact probability (or significance level) that the obtained result is purely a product of chance is calculated (Hays, 1994). The test statistic of 13.02 was highly significant ($p = 0.002$, two-tailed). Thus, we rejected the hypothesis of independence and concluded that there is a difference in the distribution of assessment of the usefulness of formal routines as an efficient medium for transfer of knowledge and experience between software developers and managers.

Table 8.5: Alternative media for knowledge transfer.

	Rank	
Medium	**Developers**	**Managers**
Experience base/newsgroups	1	1
Socialization	2	3
Discussion groups	3	2
Experience reports	4	3
On-the-job-training	5	6
Work with ext. consultants	6	-
Group meetings	7	5

Table 8.6: Do you regard written routines as an efficient medium for transfer of knowledge and experience?

	Software developers (*n* = 13)		**Managers (*n* = 10)**	
	Frequency	Percent	Frequency	Percent
Yes	-	-	7	70
Both	7	54	2	20
No	6	46	1	10

Table 8.7: Degree of involvement vs. assessment of formal routines as an efficient medium for transfer of knowledge and experience.

	Degree of involvement	
Efficient medium?	**Low**	**High**
Yes	-	7
Both	5	4
No	7	-

Since software developers had been involved in the process of introducing the routines to a much lesser extent than the managers (about 30% of the developers and 70% of the managers had actively *participated* in the introduction of routines, see *Table 8.4*), we compared the respondent's assessment of the routines with the level of involvement using Fisher's exact test (*Table 8.7*). The test statistic of 14.71 was highly significant ($p < 0.0005$, two-tailed). Thus, we concluded that there is a difference in the assessment of the usefulness of formal routines as an efficient medium for transfer of knowledge and experience with respect to the degree of involvement in the introduction process.

8.4 Main Survey: The Key Factors of Success in SPI

In this section, we report on the results from testing the hypotheses and exploring the relationships in the main, quantitative survey. First, however, we make an evaluation of the analytical assumptions underlying the analyses by examining the statistical properties of the independent and dependent variables.

8.4.1 Evaluation of analytical assumptions

Since we are testing the hypotheses in this study using parametric statistics such as the Pearson product-moment correlation coefficient, the t-test, the F statistic, and multiple regression analysis, it is essential that the assumptions of these analyses be met. Meeting these assumptions will ensure that the data obtained are truly representative and that we have obtained the best results possible. Hence, the purpose of this section is to address the assessment of both the individual variables and the overall relationships of the variate.

In the course of analyzing the bivariate correlations, calculating the regression coefficients, and estimating the dependent variable, we examined whether the fundamental assumptions underlying multiple regression analysis were met. Therefore, the following assumptions were examined: normality, homoscedasticity, linearity, and independence of the error terms (Davis, 1996). In addition, we examined potentially influential observations (outliers) and the degree of collinearity and multicollinearity.

The first and most fundamental assumption in parametric statistics and hence, in multiple regression analysis is the *normality* of the data. Two important characteristics to measure the normality of the data distributions, kurtosis and skewness, are reported in *Table 8.8*. While kurtosis is a measure of the "peakedness" or "flatness" of a data distribution compared with the normal distribution, skewness is a measure of the symmetry of a distribution (Hair *et al.*, 1998). In addition, we tested the assumption of normality more formally through use of the one-sample Kolmogorov-Smirnov test for normality, which calculates the level of significance for the difference from a normal distribution. The Kolmogorov-Smirnov Z statistic for all variables are reported in *Table 8.8* together with the corresponding p-values.

Both kurtosis and skewness should have an absolute value smaller than one (Kaplan, 1990). Another common criterion is that the ratio of each of these statistics to its standard error should be less than +2 and greater than –2 (SPSS, 1999b). Examining the values of skewness and kurtosis in *Table 8.8* gave no serious concern in this study, since their values for all scale means were relatively low (zero indicating normal spread). All ratios of kurtosis and skewness to its standard error had an absolute value smaller than 2. Hence, the skewness and kurtosis coefficients of all variables indicate support for normality.

Furthermore, the Kolmogorov-Smirnov Z was computed from the largest difference between the observed and theoretical cumulative distribution functions for each variable. The null hypothesis is that the sample distribution is normally distributed. So, if the Z statistic is significant, then the null hypothesis is rejected and the assumption of normality is not supported. None of the Kolmogorov-Smirnov Z values were significant at $p < 0.05$, but both business orientation ($Z = 1.36, p = 0.05$) and involved leadership ($Z = 1.36, p = 0.05$) showed values exactly at the 0.05 level of significance.

Table 8.8: Tests for normality of the independent and dependent variables.

Variable	**Kurtosis**	**Kurtosis/ std. error**	**Skewness**	**Skewness/ std. error**	**K-S *Z***	**Sig.**
Independent:						
1. Business orientation	.25	.56	-.29	-1.33	1.36	.05
2. Involved leadership	.14	.32	-.37	-1.68	1.36	.05
3. Employee participation	.12	.26	-.08	-.37	.97	.30
4. Measurement	-.05	-.11	.13	.57	.86	.46
5. Exploitation	.08	.18	-.40	-1.82	1.28	.08
6. Exploration	-.26	-.59	-.21	-.95	1.11	.17
Dependent:						
Overall SPI success	-.04	.78	-.33	-1.82	1.29	.07

Table 8.9: Tests of homogeneity of variances.

Independent variable	Levene Statistic	Significance
1. Business orientation	1.18	.30
2. Involved leadership	1.95	.02
3. Employee participation	1.15	.32
4. Measurement	1.52	.10
5. Exploitation	1.45	.14
6. Exploration	1.46	.12

Based on an overall assessment of the results from examining the means, standard deviations, skewness, kurtosis, and the one-sample Kolmogorov-Smirnov tests for each of the variables, we concluded that the assumption of normality is supported for all variables.

The second assumption that has to apply is *homoscedasticity*, which refers to the assumption that the dependent variable exhibits equal levels of variance across values of the independent variables. Homoscedasticity is desirable because the variance of the dependent variable being explained in the dependence relationship should not be concentrated in only a limited range of the independent values (Hair *et al.*, 1998). The most commonly used statistical test for homogeneity of variances is the Levene test of equal variances. The null hypothesis is that the error variance of the dependent variable is equal across the independent variables (SPSS, 1999b). Hence, if the Levene statistic is significant, then the null hypothesis is rejected and the assumption of homoscedasticity is not supported. According to Hair *et al.* (1998), the statistical tests for significance with the Levene statistic should be more conservative, for example using 0.01 instead of 0.05.

The analysis of variance between SPI success and each of the independent variables supports the assumption of homoscedasticity, since none of the Levene tests for equal variances, shown in *Table 8.9*, were significant at $p < 0.01$. For the involved leadership scale the significance level was 0.02, indicating that the equal variance hypothesis is rejected at the 0.05 level of significance and that SPI success exhibits different levels of variance across the range of involved leadership at this level of significance. However, this suggestion is not supported at the 0.01 level of significance. Consequently, we conclude that the assumption of homoscedasticity across the range of independent variables is supported.

The third assumption that has to apply is the *linearity* of the relationship between the dependent and independent variables. This relationship represents the degree to which the change in the dependent variable is associated with the independent variable (Hair *et al.*, 1998). Since correlations only represent the linear association between variables, any nonlinear effects will not be represented in the correlation value. Thus, a violation of the linearity assumption may result in an underestimation of the actual strength of the relationship.

We assessed linearity by an analysis of variance table testing the assumption that the mean of the dependent variable is a linear function of each of the independent variables. The null hypothesis is that the mean of the dependent variable is a linear function of the value of the independent variable (SPSS, 1999b). Hence, if the deviation from linearity is significant, then the null hypothesis is rejected and the assumption of linearity is not supported. *Table*

8.10 shows the *F* statistic and corresponding significance levels that was computed for SPI success by all independent variables.

None of the deviations from linearity were significant at the 0.01 level of significance, implying that the assumption of linearity is supported for all variables at this level. However, the relationship with the exploitation scale ($F = 1.86$; $p = .04$) was significant at the 0.05 level, rejecting the null hypothesis of linearity at this level. On the other hand, we will not use the multiple regression analysis to assess the magnitude of the regression coefficients, but only test for their significance in the regression equation in order to determine the level of support for the research hypotheses. Based on these results, we concluded that the linearity assumption was sufficiently supported. However, we should be aware of a possible underestimation of the strength of the actual relationship between the exploitation variable and SPI success.

The fourth assumption is the *independence of the error terms*, which deals with the effect of carryover from one observation to another. That is, we assume that any predicted value is not related to any other prediction. We can best identify such an occurrence by examining the pattern in the residuals (Hair *et al*., 1998). The Durbin-Watson test for serial correlation measures the association between adjacent residuals and hence, possible violations of the independence assumption. As a rule of thumb, a test statistics close to 2 indicates that there is no problem of association between adjacent residuals. The Durbin-Watson statistic in the sample was 2.09. Based on this result, we concluded that the independence assumption was supported.

Finally, we made an assessment of each observation across the set of variables in order to identify any *observations that are influential* and to determine whether they should be excluded from the analysis. Cook's distance is considered the single most representative measure of influence on overall fit in multiple regression analysis (Hair *et al*., 1998). It measures the change in the predicted values when the *i*th case is deleted, as well as the observation's distance from the other observations (leverage). A rule of thumb is to identify observations with a Cook's distance of 1.0 or greater. The highest Cook's distance in the sample was 0.20, which is considerably lower than the threshold of 1.0. Hence, no observations were excluded from the multiple regression analysis, and no further analysis of influential observations was made.

Table 8.10: Tests for linearity of the relationship between the dependent and independent variables.

Independent variable	*F* Statistic	Significance
1. Business orientation	.49	.94
2. Involved leadership	.61	.87
3. Employee participation	1.53	.10
4. Measurement	.62	.86
5. Exploitation	1.86	.04
6. Exploration	.95	.52

In any interpretation of the regression variate, we should we aware of the impact of *collinearity* and *multicollinearity*. Collinearity is the expression for the relationship between two independent variables (Hair *et al.*, 1998). A means of identifying collinearity is an examination of the correlation matrix for the independent variables. The presence of high correlation (generally those of 0.90 and above) is the first indication of substantial collinearity (Hair *et al.*, 1998, p.191).

Table 8.11 shows the means, standard deviations, and correlations among the independent variables. Out of 15 correlations between the independent variables, two have a correlation coefficient larger than 0.5. The highest correlation (0.63) is between involved leadership and business orientation. Bryman and Cramer (1997) suggested 0.80 instead of 0.90 as the threshold: "The Pearson's *r* between each pair of independent variables should not exceed 0.80; otherwise the independent variables that show a relationship at or in excess of 0.80 may be suspected of exhibiting multicollinearity" (*ibid.*, p. 257). However, all correlation coefficients in this research are below the cut-off of 0.80 for the collinearity problem.

Multicollinearity is the expression for the relationship between more than two independent variables (Hair *et al.*, 1998). That is, one of the independent variables is a linear combination of other independent variables. Two common measures for assessing multicollinearity are the *tolerance* value and its reciprocal – the *variance inflation factor* (VIF). Tolerance is the amount of variability of the selected independent variable not explained by the other independent variables (Hair *et al.*, 1998). A variable with very low tolerance contributes little information to a model and can cause computational problems. Thus, small tolerance values (and correspondingly high VIF values) denote high multicollinearity.

A common cut-off threshold is a tolerance value of 0.10, and thus a VIF value of 10.0 (Hair *et al.*, 1998). This cutoff value corresponds to a multiple correlation of 0.95. However, restricting the multiple correlation to 0.90, similar to the rule applied for the pairwise correlation matrix, would result in a tolerance value of 0.19. Thus any variables with tolerance values below 0.19 (or a VIF above 5.3) would have a multiple correlation of more than 0.90.

Table 8.11: Item means, standard deviations and correlations among the independent variables.

Independent variable	**Mean**	**S.D.**	**1**	**2**	**3**	**4**	**5**	**6**
1. Business orientation	3.27	0.66	1.00					
2. Involved leadership	3.52	0.69	0.63***	1.00				
3. Employee participation	3.49	0.52	0.39***	0.37***	1.00			
4. Measurement	3.26	0.61	0.45***	0.41***	0.24**	1.00		
5. Exploitation	3.34	0.62	0.57***	0.46***	0.47***	0.41***	1.00	
6. Exploration	3.43	0.54	0.17*	0.11	0.46***	0.13	0.18*	1.00

* $p < 0.05$ ** $p < 0.005$ *** $p < 0.0005$

1. All *t*-tests are one-tailed.

Table 8.12: Tests for multicollinearity between the independent variables.

Independent variable	R	Tolerance	Variance Inflation Factor
1. Business orientation	.72	.48	2.08
2. Involved leadership	.66	.56	1.79
3. Employee participation	.62	.61	1.64
4. Measurement	.51	.74	1.34
5. Exploitation	.65	.58	1.73
6. Exploration	.46	.78	1.28

To investigate the potential problem of multicollinearity, we regressed each independent variable against the remaining independent variables. The multiple correlation coefficients resulting from these regressions, shown in *Table 8.12*, are high, but they do not represent multicollinearity problems since they are all below 0.90. The highest VIF value in the sample was 2.08. These results indicate that interpretation of the regression variate coefficients should not be affected adversely by collinearity and multicollinearity in this research.

In conclusion, kurtosis, skewness, and the one-sample Kolmogorov-Smirnov tests are within the acceptable range for the normal distribution assumption. The assumptions of homoscedasticity, linearity, and independence of the error terms were supported, and no influential observations were identified. Furthermore, investigations of collinearity and multicollinearity indicate no problems. Hence, the investigations of the statistical properties of the independent and dependent variables do not cause concern for testing the hypotheses.

With the sample deemed adequate for the objectives of the study, and the assumptions assessed for the independent and dependent variables, we now proceed to testing the hypotheses.

8.4.2 *Testing individual relationships*

Hypotheses 1 through 6 consider the individual relationships between SPI success and each of the six independent variables. The testing of these hypotheses calls for the use of bivariate correlations. In addition to examine each independent variable's correlation with overall SPI success, we also examined the correlations with each of the two underlying success measures: perceived level of success and organizational performance. However, appropriate tests of the bivariate correlations require that the two contextual factors, environment and organizational size, be partialled from the analysis. *Table 8.13* shows both the zero-order correlations and the partial correlations when the effects of the contextual variables have been removed between the independent variables and each of the SPI success measures.

Hypothesis 1. Based on the discussion in *Chapter 5*, we defined business orientation as *the extent to which SPI goals and actions are aligned with explicit and implicit business goals and strategies*. Furthermore, we argued that business orientation is a key factor to facilitate a successful SPI program. *Ceteris paribus*, hypothesis 1: SPI success is positively associated with business orientation.

Table 8.13: Tests of hypotheses H1 – H6.

	Overall SPI success		Perceived level of success		Organizational performance	
Variable (*N* = 120)	***r***	***pr***	***r***	***pr***	***r***	***pr***
1. Business Orientation	.62***	.61***	.59***	.59***	.46***	.44***
2. Involved leadership	.55***	.54***	.58***	.57***	.34***	.34***
3. Employee participation	.55***	.59***	.49***	.52***	.43***	.49***
4. Measurement	.50***	.48***	.44***	.44***	.40***	.38***
5. Exploitation	.59***	.59***	.55***	.56***	.44***	.44***
6. Exploration	.21*	.25**	.15*	.18*	.20*	.25**

* $p < 0.05$ ** $p < 0.005$ *** $p < 0.0005$

1. All *t*-tests are one-tailed.
2. Partial *r* (*pr*) is the correlation between the success measure and the independent variable when the contextual factors (environment and organizational size) are held constant.

Hypothesis 1 can be stated in the null and alternate as follows:

$H1_0$: There is no relationship between business orientation and SPI success.

This is statistically expressed by $H1_0$: $\rho = 0$

where ρ represents the correlation between business orientation and SPI success.

$H1_A$: There will be a positive correlation between business orientation and SPI success.

This is statistically expressed by $H1_A$: $\rho > 0$

The zero-order Pearson correlation results between business orientation and overall SPI success showed a positive and highly significant correlation coefficient of 0.62 ($p < 0.0005$). The zero-order correlations with perceived level of success ($r = 0.59$, $p < 0.0005$) and organizational performance ($r = 0.46$, $p < 0.0005$) were also positive and highly significant. Furthermore, all partial correlations were positive and significant, ranging from $pr = 0.44$ ($p < 0.0005$) to $pr = 0.61$ ($p < 0.0005$). Since all correlations were positive and highly significant, the null hypothesis was rejected and the alternate hypothesis accepted. That is, *business orientation is positively associated with SPI success*.

Hypothesis 2. Based on the discussion in *Chapter 5*, we defined involved leadership as *the extent to which leaders at all levels in the organization are genuinely committed to and actively participates in SPI.* Furthermore, we argued that involved leadership is a key factor to facilitate a successful SPI program. *Ceteris paribus*, hypothesis 2: SPI success is positively associated with involved leadership.

Hypothesis 2 can be stated in the null and alternate as follows:

$H2_0$: There is no relationship between involved leadership and SPI success.

This is statistically expressed by $H2_0$: $\rho = 0$

where ρ represents the correlation between involved leadership and SPI success.

$H2_A$: There will be a positive correlation between involved leadership and SPI success.

This is statistically expressed by $H2_A$: $\rho > 0$

The zero-order Pearson correlation results between involved leadership and overall SPI success showed a positive and highly significant correlation coefficient of 0.55 ($p < 0.0005$). The zero-order correlations with perceived level of success ($r = 0.58$, $p < 0.0005$) and organizational performance ($r = 0.34$, $p < 0.0005$) were also positive and highly significant. Furthermore, all partial correlations were positive and significant, ranging from $pr = 0.34$ ($p < 0.0005$) to $pr = 0.57$ ($p < 0.0005$). Since all correlations were positive and highly significant, the null hypothesis was rejected and the alternate hypothesis accepted. That is, *involved leadership is positively associated with SPI success*.

Hypothesis 3. Based on the discussion in *Chapter 5*, we defined employee participation as *the extent to which employees use their knowledge and experience to decide, act and take responsibility for SPI*. Furthermore, we argued that employee participation is a key factor to facilitate a successful SPI program. *Ceteris paribus*, hypothesis 3: SPI success is positively associated with employee participation.

Hypothesis 3 can be stated in the null and alternate as follows:

$H3_0$: There is no relationship between employee participation and SPI success.

This is statistically expressed by $H3_0$: $\rho = 0$

where ρ represents the correlation between employee participation and SPI success.

$H3_A$: There will be a positive correlation between employee participation and SPI success.

This is statistically expressed by $H3_A$: $\rho > 0$

The zero-order Pearson correlation results between employee participation and overall SPI success showed a positive and highly significant correlation coefficient of 0.55 ($p < 0.0005$). The zero-order correlations with perceived level of success ($r = 0.49$, $p < 0.0005$) and organizational performance ($r = 0.43$, $p < 0.0005$) were also positive and highly significant. Furthermore, all partial correlations were positive and significant, ranging from $pr = 0.49$ ($p < 0.0005$) to $pr = 0.59$ ($p < 0.0005$). Since all correlations were positive and highly significant, the null hypothesis was rejected and the alternate hypothesis accepted. That is, *employee participation is positively associated with SPI success*.

Hypothesis 4. Based on the discussion in *Chapter 5*, we defined concern for measurement as *the extent to which the software organization collects and utilizes quality data to guide and assess the effects of SPI activities*, and argued that concern for measurement is a key factor to facilitate a successful SPI program. *Ceteris paribus*, hypothesis 4: SPI success is positively associated with the use of measurement.

Hypothesis 4 can be stated in the null and alternate as follows:

$H4_0$: There is no relationship between concern for measurement and SPI success.

This is statistically expressed by $H4_0$: $\rho = 0$

where ρ represents the correlation between concern for measurement and SPI success.

$H4_A$: There will be a positive correlation between concern for measurement and SPI success.

This is statistically expressed by $H4_A$: $\rho > 0$

The zero-order Pearson correlation results between concern for measurement and overall SPI success showed a positive and highly significant correlation coefficient of 0.50 ($p < 0.0005$). The zero-order correlations with perceived level of success ($r = 0.44$, $p < 0.0005$) and organizational performance ($r = 0.40$, $p < 0.0005$) were also positive and highly significant. Furthermore, all partial correlations were positive and significant, ranging from $pr = 0.38$ ($p < 0.0005$) to $pr = 0.48$ ($p < 0.0005$). Since all correlations were positive and highly significant, the null hypothesis was rejected and the alternate hypothesis accepted. That is, *concern for measurement is positively associated with SPI success*.

Hypothesis 5. In *Chapter 5*, we defined learning strategy as *the extent to which a software organization is engaged in the exploitation of existing knowledge and in the exploration of new knowledge*. Furthermore, we argued that balancing the refinement of the existing skill base, with the experimentation of new ideas are important in order to find alternatives that improve on old ones. *Ceteris paribus*, hypothesis 5: SPI success is positively associated with exploitation of existing knowledge.

Hypothesis 5 can be stated in the null and alternate as follows:

$H5_0$: There is no relationship between exploitation of existing knowledge and SPI success.

This is statistically expressed by $H5_0$: $\rho = 0$

where ρ represents the correlation between exploitation of existing knowledge and SPI success.

$H5_A$: There will be a positive correlation between exploitation of existing knowledge and SPI success.

This is statistically expressed by $H5_A$: $\rho > 0$

The zero-order Pearson correlation results between exploitation of existing knowledge and overall SPI success showed a positive and highly significant correlation coefficient of 0.59 ($p < 0.0005$). The zero-order correlations with perceived level of success ($r = 0.55$, $p < 0.0005$) and organizational performance ($r = 0.44$, $p < 0.0005$) were also positive and highly significant. Furthermore, all partial correlations were positive and significant, ranging from $pr = 0.44$ ($p < 0.0005$) to $pr = 0.59$ ($p < 0.0005$). Since all correlations were positive and highly significant, the null hypothesis was rejected and the alternate hypothesis accepted. That is, *exploitation of existing knowledge is positively associated with SPI success*.

Hypothesis 6. In addition to hypothesis 5 and the argument concerning the balance between the refinement of the existing skill base with the experimentation of new ideas, a second hypothesis was defined regarding learning strategy; *Ceteris paribus*, hypothesis 6: SPI success is positively associated with exploration of new knowledge.

Hypothesis 6 can be stated in the null and alternate as follows:

$H6_0$: There is no relationship between exploration of new knowledge and SPI success.

This is statistically expressed by $H6_0$: $\rho = 0$

where ρ represents the correlation between exploration of new knowledge and SPI success.

$H6_A$: There will be a positive correlation between exploration of new knowledge and SPI success.

This is statistically expressed by $H6_A$: $\rho > 0$

The zero-order Pearson correlation results between exploration of new knowledge and overall SPI success showed a positive and significant correlation coefficient of 0.21 ($p < 0.05$). The zero-order correlations with perceived level of success ($r = 0.15$, $p < 0.05$) and organizational performance ($r = 0.20$, $p < 0.05$) were also positive and significant. Furthermore, all partial correlations were positive and significant, ranging from $pr = 0.18$ ($p < 0.05$) to $pr = 0.25$ ($p < 0.005$). Since all correlations were positive and significant, the null hypothesis was rejected and the alternate hypothesis accepted. That is, *exploration of new knowledge is positively associated with SPI success*.

Taken together, all zero-order and partial correlations involved in testing hypotheses 1 through 6 were significant and in the hypothesized directions. This indicates support for the validity of all major measures used in this research. Thus, the findings in *Table 8.13* support Hypotheses 1 through 6, along with the underlying assumption that SPI provides increased levels of performance to the software company.

8.4.3 *Testing overall relationships*

Besides examining the independent contributions of each of the independent variables, as represented by hypotheses 1 through 6, we also examined the joint contribution of the independent variables to the explanation of SPI success. Furthermore, our aim was also to consider their contribution to the variate and its predictions in order to identify the critical factors for SPI success.

***Hypothesis** 7*. Hypothesis 7 can be stated in the null and alternate as follows:

$H7_0$: The six independent variables in the theoretical framework will not significantly explain the variance in SPI success.

This is statistically expressed by $H7_0$: $R^2 = 0$

where R^2 (R-square) represents the proportion of variance in SPI success that can be accounted for by the six independent variables.

$H7_A$: The six independent variables of business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge will significantly explain a large amount of variance in SPI success.

A "large" amount is defined as a large effect size, $f^2 \geq 0.35$, according to Cohen's (1988, 1992) categorization (see *Chapter 6* for more details regarding operationalization of effect sizes), where

$$f^2 = \frac{R^2}{1-R^2} \qquad (8.1)$$

Thus, in terms of the effect size, the squared multiple correlation can be expressed as

$$R^2 = \frac{f^2}{1+f^2} \qquad (8.2)$$

Given this relationship between effect size (f^2) and the squared multiple correlation (R^2), a large effect size of $f^2 \geq 0.35$ corresponds to a squared multiple correlation of $R^2 \geq 0.26$ and a multiple correlation coefficient of $R \geq 0.51$.

We applied multiple regression analysis by forming the variate of independent variables to test this hypothesis. The model summary of the multiple regression analysis between the six independent variables and the dependent variable is shown in *Table 8.14*, the analysis of variance in *Table 8.15*, and the regression coefficients in *Table 8.16*. The multiple correlation coefficient R of the six independent variables with the dependent variable was 0.76 and the adjusted R-square was 0.56 (as compared to a coefficient of multiple determination – R-square – of 0.58), thus, explaining 56 percent of the variation in SPI success. Furthermore, the

F-value of 25.95 was highly significant ($p < 0.0005$), indicating that there is a strong relationship between the six independent variables and SPI success.

In other words, the results indicate that a large amount (56 %) of the variance (adjusted *R*-square) in SPI success has been highly significantly ($p < 0.0005$) explained by the six independent variables. Hence, the null hypothesis was rejected and the alternate hypothesis accepted. That is, *the six independent variables of business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge significantly explain a large amount of the variance in SPI success*.

However, the above result only applies to sample values. Therefore, in order to make inferences about the multiple correlation, we used Konishi's (1981) extension of the Fisher *r*-to-*Z* transformation to test the exact hypothesis that the population value of *R* is large. That is, we tested the modified null hypothesis

$H7_{\mathrm{M0}}$: $\rho < 0.51$

against the alternative

$H7_{\mathrm{MA}}$: $\rho \geq 0.51$

The test statistic, C_R, is referred to the normal distribution (Konishi, 1981), and given by

$$C_R = \left\{ Z_R - \zeta_\rho - \frac{1}{2\rho(N-1)}\left(K - 1 + \rho^2\right) \right\} \sqrt{N-1} \qquad (8.3)$$

in which, Z_R is the Fisher *Z* value corresponding to the sample *R* value, ζ_ρ is the Fisher *Z* value corresponding to the population ρ value stated in the null hypothesis, *K* is the number of predictor variables, and *N* is the sample size.

The test statistic for the modified null hypothesis with $K = 6$ predictor variables, a sample size of $N = 120$, and a sample multiple correlation of $R = 0.76$ was $C_R = 4.26$. In a standard normal distribution, a value of 4.26 is highly significant and thus falls outside the acceptance region for the $\alpha = 0.05$ level. Hence, in light of the sample evidence we rejected the null hypothesis that the true value of the multiple correlation is less than 0.51, and accepted the alternate hypothesis that *the six predictor variables significantly* ($p < 0.00005$) *explain a large amount* ($\rho \geq 0.51$) *of the variance in SPI success*.

The contributions of each independent variable to the variate, however, differ substantially. As can be seen in *Table 8.16*, two of the independent variables – *involved leadership* and *exploration of new knowledge* – did not contribute significantly to the explanation of SPI success. Of the remaining four significant variables, *employee participation* seems to be associated with the highest explanatory power, since it achieved the highest standardized regression coefficient ($\beta = 0.297$, $t = 3.80$, $p < 0.0005$). Next came *business orientation* ($\beta = 0.245$, $t = 2.78$, $p < 0.01$), followed by *concern for measurement* ($\beta = 0.200$, $t = 2.83$, $p < 0.01$) and, finally, *exploitation of existing knowledge* ($\beta = 0.177$, $t = 2.20$, $p < 0.05$). It is important to note, however, that the weights the independent variables carry in the multiple regression equation are always relative to the entire set of predictors employed (Hays, 1994). That is, the predictive strength that some independent variable seems to show for the dependent variable may differ if the set of remaining variables is altered. To further interpret the individual contributions to the variate and determine the critical factors of SPI success, we applied the stepwise regression procedure, which is described in the next section.

The two contextual factors were added to the model to test the sensitivity of the variate to organizational size and environmental turbulence. The estimated coefficients of the independent variables of the original model were not significantly changed by the introduction of the contextual factors. *The results hold for small, as well as large, software organizations, and for organizations operating in stable as well as in turbulent environments.* These results support the robustness of the original regression model results. Furthermore, the model with the two contextual factors added (R-square = 0.59, adjusted R-square = 0.56, F = 19.60, $p < 0.0005$) did not explain significantly more variance as compared with the original model (R-square = 0.58, adjusted R-square = 0.56, $F = 25.95$, $p < 0.0005$).

Table 8.14: Multiple regression model summary using all independent variables.

R	***R*-square**	**Adjusted *R*-square**	**Std. Error of the estimate**
.761	.579	.557	1.402

Table 8.15: Multiple regression analysis of variance using all independent variables.

	Sum of squares	***Df***	**Mean square**	***F* ratio**
Regression	305.816	6	50.969	25.949***
Residuals	221.958	113	1.964	

*** $p < 0.0005$

Table 8.16: Multiple regression between all independent and dependent variables.

	Unstandardized coefficients		**Standardized coefficients**	
Variables	***B***	**Std. error**	***Beta***	***t*-value**
1. Business Orientation	.157	.056	.245	2.778**
2. Involved leadership	.076	.050	.125	1.526
3. Employee participation	.173	.045	.297	3.803***
4. Measurement	.115	.041	.200	2.828**
5. Exploitation	.121	.055	.177	2.201*
6. Exploration	-.019	.034	-.039	-.559

* $p < 0.05$ ** $p < 0.01$ *** $p < 0.0005$

8.4.4 Examining individual contributions to the variate

To examine the contributions of each independent variable to the variate, we applied stepwise regression with forward selection. With this procedure we started by adding the independent variable with the greatest contribution. Additional independent variables were selected in terms of the incremental explanatory power they could add to the regression model. Independent variables were added as long as their partial correlation coefficients were statistically significant. Furthermore, independent variables could also be dropped if their predictive power dropped to a nonsignificant level when another independent variable was added to the model. We used conventional stepwise criteria, i.e. the probability of F-to-enter was ≤ 0.05, and the probability of F-to-remove was ≥ 0.10. That is, the squared part correlation (or gain in SPI success variance accounted for) had to be significant at or beyond the 0.05 level for any new variable to be selected for inclusion in the regression equation.

Using the stepwise procedure, four of the six independent variables were included in the final model. They were entered into the equation in the following order: *business orientation*, *employee participation*, *concern for measurement*, and *exploitation of existing knowledge*. Thus, the final model included the same four variables that were significant in the full regression model.

The stepwise regression model summary in *Table 8.17* shows that R-square for the final model is 0.57 and adjusted R-square is 0.55. These values are comparable to the full regression model with all six independent variables that showed an R-square value of 0.58 and adjusted R-square of 0.56 (see *Table 8.14*). In other words, the joint predictive power of the four independent variables resulting from the stepwise regression included in model D, is comparable to the predictive power of the full regression model, which includes all six independent variables.

The standardized stepwise regression coefficients are reported in *Table 8.19*. Of the four independent variables included in the final model, *business orientation* seems to be associated with the highest explanatory power since it achieved the highest standardized regression coefficient ($\beta = 0.304$, $t = 3.86$, $p < 0.0005$). Next came *employee participation* ($\beta = 0.293$, $t = 4.17$, $p < 0.0005$), followed by *concern for measurement* ($\beta = 0.214$, $t = 3.06$, $p < 0.005$) and, finally, *exploitation of existing knowledge* ($\beta = 0.190$, $t = 2.37$, $p < 0.05$). Compared to the results from the full regression model with all six independent variables, the predictive strength of each variable have not been considerably altered, although the rank order between the two variables with the highest explanatory power has changed. However, as we made clear in the preceding section, the weights the independent variables carry in the multiple regression equation are always relative to the entire set of predictors employed.

The results of the stepwise regression confirm the full multiple regression analysis regarding the contributions of each independent variable to the variate. Hence, given the original set of six independent variables, the four variables *employee participation*, *business orientation*, *concern for measurement*, and *exploitation of existing knowledge* can be considered as *critical factors for SPI success*.

In *Section 8.1.1* we found that multicollinearity should not have a substantial impact on the estimated regression variate. However, the results of the multiple regression analysis show that it, nevertheless, does have an impact on the composition of the variate, and that correlations among the independent variables may make some variables redundant in the predictive effort, which is the case for *involved leadership* and *exploration of new knowledge*.

However, as shown in *Section 8.1.2*, this does not reflect their individual relationships with the dependent variable, but instead indicates that in a multivariate context, they are not needed together with the remaining set of four independent variables to explain the variance in SPI success. Therefore, we cannot determine the importance of independent variables based solely on the derived variate, since relationships among the independent variables may "mask" relationships that are not needed for predictive purposes, but represent substantive explanatory findings nonetheless.

Table 8.17: Stepwise regression model summary – final model.

Model	***R***	***R*-square**	**Adjusted *R*-square**	**Std. error of the estimate**
A	.623	.388	.383	1.654
B	.707	.500	.492	1.501
C	.740	.548	.536	1.434
D	.754	.569	.554	1.406

Model A: business orientation.
Model B: business orientation, participation.
Model C: business orientation, participation, measurement.
Model D: business orientation, participation, measurement, exploitation.

Table 8.18: Stepwise regression analysis of variance.

Model		**Sum of squares**	***Df***	**Mean square**	***F* ratio**
A	Regression	204.939	1	204.939	74.907***
	Residuals	322.835	118	2.736	
B	Regression	264.043	2	132.021	58.569***
	Residuals	263.732	117	2.254	
C	Regression	289.214	3	96.405	46.877***
	Residuals	238.560	116	2.057	
D	Regression	300.301	4	75.075	37.955***
	Residuals	227.473	115	1.978	

*** $p < 0.0005$
Model A: business orientation.
Model B: business orientation, participation.
Model C: business orientation, participation, measurement.
Model D: business orientation, participation, measurement, exploitation.

Table 8.19: Stepwise regression coefficients.

	Unstandardized coefficients		Standardized coefficients	
Variables	***B***	**Std. error**	***Beta***	***t*-value**
1. Business Orientation	.195	.051	.304	3.858***
3. Employee participation	.170	.041	.293	4.168***
4. Measurement	.123	.040	.214	3.055**
5. Exploitation	.130	.055	.190	2.367*

* $p < 0.05$ ** $p < 0.005$ *** $p < 0.0005$

Table 8.20: Results of assessing the study's research hypotheses.

Hypothesis	Result	Comments
1	Strong support	Both zero-order and partial bivariate correlations are highly significant and the multiple regression coefficient is also highly significant.
2	Partial support	Zero-order and partial bivariate correlations are highly significant. However, the multiple regression coefficients for the variable is not significant.
3	Strong support	Both zero-order and partial bivariate correlations are highly significant and the multiple regression coefficient is also highly significant.
4	Strong support	Both zero-order and partial bivariate correlations are highly significant and the multiple regression coefficient is also highly significant.
5	Strong support	Both zero-order and partial bivariate correlations are highly significant and the multiple regression coefficient is also highly significant.
6	Partial support	Zero-order and partial bivariate correlations are significant. However, the multiple regression coefficient for the variable is not significant.
7	Strong support	Both multiple regression and stepwise regression analysis show that the set of six independent variables explain a highly significant and high percentage of the variance in SPI success.

In conclusion, both the multiple regression analysis and the stepwise regression analysis have given additional support for hypotheses 1, 3, 4, and 5. Furthermore, the stepwise regression analysis has also given additional support for hypothesis 7. However, the regression analyses did not support hypotheses 2 and 6, provided that all hypotheses are considered as one set. In other words, considering both bivariate correlation analyses and multiple regression analyses, we find strong support for hypotheses 1, 3, 4, 5, and 7, and partial support for hypotheses 2 and 6 (see *Table 8.20*).

8.4.5 *Exploring the effects of environmental conditions*

To examine whether a software organization's environmental conditions affects its success factors and the resulting performance, we compared the effects of environmental stability or turbulence. As explained in *Section 6.6*, we defined contrasted criterion groups based on the upper and lower third of the sample distribution. Thus, stability was defined as the lower third (N = 40) and turbulence was defined as the upper third (N = 44) of the environment distribution.

We examined the effects of environmental conditions by comparing the differences of the means between organizations operating in stable and turbulent environments using two-tailed *t*-tests. Results of these tests are summarized in *Table 8.21*. For each construct the table provides the mean score, standard deviation, and *t*-value.

No statistically significant differences between stable and turbulent environments were observed for the dependent or independent variables. To shed additional light on this finding, we subsequently compared the level of turbulence between large and small organizations, and also between successful and less successful organizations. However, there was no statistically significant difference, neither in the mean level of environmental turbulence between large and small organizations (*Table 8.22*) nor between successful and less successful organizations (*Table 8.23*).

Table 8.21: Stable versus turbulent environment.

	Stable (*n* = 40)		**Turbulent (*n* = 44)**		
	Mean	**S.D.**	**Mean**	**S.D.**	***t*-value**
SPI success factors:					
1. Business Orientation	3.26	.59	3.23	.82	.18
2. Involved leadership	3.36	.69	3.59	.72	-1.50
3. Employee participation	3.42	.54	3.59	.49	-1.53
4. Measurement	3.24	.62	3.36	.62	-.93
5. Exploitation	3.33	.64	3.35	.67	-.14
6. Exploration	3.43	.56	3.47	.57	-.30
Performance:					
Overall SPI success	3.44	.40	3.52	.60	-.65

1. All *t*-tests are two-tailed.

Table 8.22: Level of turbulence in large versus small software organizations.

	Large (*n* = 44)		**Small (*n* = 45)**		
	Mean	**S.D.**	**Mean**	**S.D.**	***t*-value**
Environmental conditions	4.11	1.19	4.17	1.41	-.19

1. Two-tailed *t*-test.

Table 8.23: Level of turbulence in successful versus less successful software organizations.

	Successful (*n* = 39)		Less successful (*n* = 41)		
	Mean	**S.D.**	**Mean**	**S.D.**	***t*-value**
Environmental conditions	4.35	1.43	4.04	1.23	1.04

1. Two-tailed *t*-test.

8.4.6 Exploring the effects of organizational size

To examine whether a software organization's size affects its success factors and the resulting SPI success, we compared small (≤ 30 developers, $N = 45$) and large (≥ 200 developers, $N = 44$) organizations. The effects of organizational size were examined by testing the differences of the means between the small and large organizations using two-tailed *t*-tests. Results of these tests are summarized in *Table 8.24*. For each construct the table provides the mean score, standard deviation, and *t*-value.

Five of the six independent variables showed a statistically significant difference between large and small software organizations. Large organizations reported a significantly greater extent of *business orientation* ($t = 2.71$, $p < 0.01$), *concern for measurement* ($t = 2.61$, $p < 0.05$), and *involved leadership* ($t = 1.79$, $p < 0.1$) than small organizations. However, small organizations reported a highly significantly greater extent of *exploration of new knowledge* ($t = -3.00$, $p < 0.005$) and a significantly greater extent of *employee participation* ($t = -1.75$, $p < 0.1$) than large organizations. The last independent variable; the extent of *exploitation of existing knowledge*, were similar in large and small organizations. Furthermore, with respect to organizational performance, large software organizations reported a significantly higher overall SPI success ($t = 2.18$, $p < 0.05$) than small organizations.

To further examine the effects of organizational size, we compared the responses from large successful software organizations with the responses from large less successful organizations. These results are tabulated in *Table 8.25*. The comparison revealed that with the exception of exploration of new knowledge, large successful organizations reported executing all of the activities defined by the independent variables to a significantly greater extent than large less successful organizations.

Similarly, we made a comparison between small successful software organizations and small less successful organizations. As shown in *Table 8.26*, small successful organizations reported executing all of the activities defined by the independent variables to a highly significantly ($p < 0.0005$) greater extent than small less successful organizations. Furthermore, comparing these results with the results of the large organizations, we observe that the differences of the means between the large organizations are smaller than the corresponding differences of the means between the small organizations.

Finally, we compared small and large software organizations that were successful. *Table 8.27* shows the differences that appear to exist between small and large organizations having SPI success. As can be seen, both small and large successful organizations reported the same level of success. Statistically significant differences were observed for two of the six independent variables: Small successful organizations reported higher levels of employee participation (t = -2.44, $p < 0.05$) and exploration of new knowledge (t = -3.50, $p < 0.001$) than the larger organizations.

Table 8.24: Large versus small software organizations.

	Large (n = 44)		Small (n = 45)		
	Mean	**S.D.**	**Mean**	**S.D.**	***t*-value**
SPI success factors:					
1. Business Orientation	3.48	.53	3.10	.77	2.71**
2. Involved leadership	3.68	.68	3.40	.75	1.79¤
3. Employee participation	3.41	.50	3.60	.49	-1.75¤
4. Measurement	3.42	.53	3.10	.61	2.61*
5. Exploitation	3.39	.55	3.30	.63	.67
6. Exploration	3.26	.55	3.59	.51	-3.00***
Performance:					
Overall SPI success	3.63	.52	3.39	.51	2.18*

¤ $p < 0.1$ * $p < 0.05$ ** $p < 0.01$ *** $p < 0.005$

1. All t-tests are two-tailed.

Table 8.25: Large successful versus large less successful software organizations.

	Large successful (n = 20)		Large less successful (n = 11)		
	Mean	**S.D.**	**Mean**	**S.D.**	***t*-value**
SPI success factors:					
1. Business Orientation	3.80	.46	3.16	.45	3.71**
2. Involved leadership	3.97	.62	3.49	.71	1.95#
3. Employee participation	3.69	.49	3.12	.49	3.11*
4. Measurement	3.72	.49	3.23	.41	2.82¤
5. Exploitation	3.71	.42	2.93	.42	4.97***
6. Exploration	3.39	.50	3.12	.52	1.44

$p = 0.06$ ¤ $p < 0.01$ * $p < 0.005$ ** $p < 0.001$ *** $p < 0.0005$

1. All t-tests are two-tailed.

Table 8.26: Small successful versus small less successful software organizations.

	Small successful (n = 12)		**Small less successful (n = 17)**		
	Mean	**S.D.**	**Mean**	**S.D.**	***t*-value**
SPI success factors:					
1. Business Orientation	3.85	.47	2.48	.60	6.63***
2. Involved leadership	4.10	.34	2.88	.58	6.49***
3. Employee participation	4.08	.37	3.20	.37	6.36***
4. Measurement	3.65	.60	2.84	.50	3.97***
5. Exploitation	3.87	.45	2.87	.52	5.38***
6. Exploration	4.01	.46	3.29	.46	4.12***

*** $p < 0.0005$

1. All *t*-tests are two-tailed.

Table 8.27: Large successful versus small successful software organizations.

	Large successful (n = 20)		**Small successful (n = 12)**		
	Mean	**S.D.**	**Mean**	**S.D.**	***t*-value**
SPI success factors:					
1. Business Orientation	3.80	.46	3.85	.47	-.29
2. Involved leadership	3.97	.62	4.10	.34	-.67
3. Employee participation	3.69	.49	4.08	.37	-2.44*
4. Measurement	3.72	.49	3.65	.60	.33
5. Exploitation	3.71	.42	3.87	.45	-1.00
6. Exploration	3.39	.50	4.01	.46	-3.50**
Performance:					
Overall SPI success	4.07	.29	3.97	.14	1.11

* $p < 0.05$ ** $p < 0.001$

1. All *t*-tests are two-tailed.

8.4.7 *Exploring the relationships between organizational context and modes of learning*[4]

To examine the relationships between organizational context and modes of learning, we studied the balance between exploitation of existing knowledge and exploration of new knowledge. Specifically, we studied how organizational size and environmental turbulence affected this balance (see Dybå, 2000b). Effects of organizational size were compared by contrasting large organizations with small organizations. Similarly, the effects of environmental turbulence were examined by contrasting stable environments with turbulent environments.

As explained in *Chapter 6*, we defined contrasted criterion groups based on the upper and lower third of the sample distribution. Thus, stability was defined as the lower third ($N = 40$) and turbulence was defined as the upper third ($N = 44$) of the environment distribution. Likewise, large organizations consisted of the upper third (≥ 200 developers, $N = 44$) and small organizations of the lower third (≤ 30 developers, $N = 45$) of the organization size distribution.

The effects of organizational size and environmental conditions on the learning modes were examined by comparing the estimated marginal means of exploitation of existing knowledge and exploration of new knowledge, respectively, using two-way analysis of variance (ANOVA) with organizational size and environment as the two classification variables.

Results showed that small software organizations kept the same level of exploitation both in stable and turbulent environments. However, they engaged in significantly ($p < 0.05$) more exploration in turbulent environments than they did in stable environments. It is important to note that the increased level of exploration did not drive out exploitation. In other words, we found support for the proposition that increased environmental turbulence requires increased levels of improvisation.

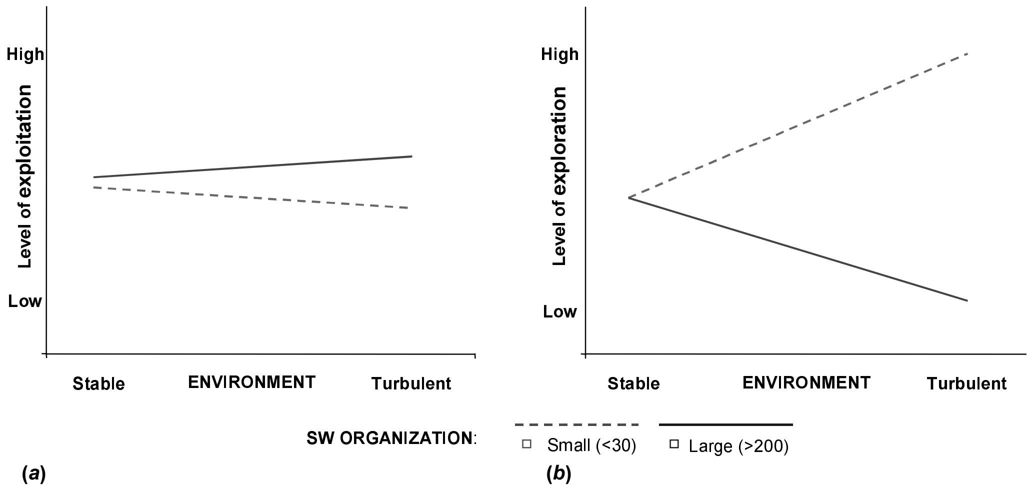


Figure 8.5: Improvement strategy versus organizational size and environmental turbulence for (*a*) exploitation and (*b*) exploration in small and large organizations.

[4] The results reported in this section are based on (Dybå, 2000b).

Similar to the small organizations, large software organizations did not differ significantly in their level of exploitation between stable and turbulent environments. In contrast to the small organizations, however, increased turbulence did not lead to increased levels of exploration. On the contrary, the larger organizations seemed to lower their levels of exploration during turbulent environments. However, this difference was not significant.

It is interesting to note that, as there was no difference in the level of exploitation between small and large organizations regardless of the environment, there was a marked difference in the level of exploration (see *Figure 8.5b*). The results showed that small software organizations engaged in significantly ($p < 0.001$) more exploration in turbulent environments than large software organization.

These results supports the assertion that small software organizations in turbulent environments, require improvement strategies that are more closely aligned with explorative behavior, while at the same time promoting the exploitation of past experiences. As discussed in *Chapter 4* and also in Dybå (2000b), this is at the heart of an improvement strategy based on improvisation.

8.4.8 Exploring the effects of other organizational characteristics

In addition to examining the effects of environmental turbulence and organizational size on SPI success, we also examined the effects of some other characteristics of the respondent's organizations. These characteristics were industry sector, product business, and whether the organization had a quality system in use or not. The effects of these organizational characteristics were examined by testing the frequency distributions of the successful and less successful organizations using Pearson chi-square tests of association. Results of these tests are summarized in *Table 8.28*. For each characteristic the table provides the frequency, percent, and the Pearson chi-square statistic.

The null hypothesis for the Pearson chi-square test of association is that the row and column variables are independent of each other. By definition, two table variables are *independent* if the probability that a case falls in a specific cell is the product of its marginal probabilities (SPSS, 1999b). The computed chi-square statistic for the industry sector part of *Table 8.28* was statistically significant ($\chi^2 = 8.81$, $df = 3$, $p < 0.05$). However, a rule of thumb is that for contingency tables with more than a single degree of freedom, no more than 20% of the cells should have an expected frequency of less than 5 (Hays, 1994). Clearly, this rule is violated since 50% of the cells have expected frequency less than 5. A normal practice for dealing with this problem is to pool categories in order to attain larger expected frequencies. However, we found no natural categories on which to base such a pooling. Consequently, we cannot reject the hypothesis of independence. In other words, despite the statistically significant chi-square value, we do not have enough evidence to support the claim that industry sector and SPI success are associated.

Table 8.28: Successful versus less successful software organizations.

	Successful (n = 39)		Less successful (n = 41)		
	Frequency	**Percent**	**Frequency**	**Percent**	χ^2
Industry sector:					8.81*
Public sector	-	-	5	12.2	
Banking/finance	5	12.8	3	7.3	
Manufacturing	11	28.2	4	9.8	
IT sector	20	51.3	22	53.7	
Other	3	7.7	7	17.1	
Product business:					.34
Standard applications	9	23.1	10	24.4	
Tailor made solutions	30	76.9	31	75.6	
Quality system in use:					8.92**
Yes	33	84.6	19	46.3	
No	6	15.4	22	53.7	

* $p < 0.05$ ** $p < 0.005$

1. All chi-square tests are two-tailed.
2. For industry sector, 4 cells (50%) have expected frequency less than 5.

Similarly as for industry sector, we checked if SPI success vary by the main type of product business (i.e. standard applications – “shelfware” – for the mass market versus tailor made solutions for individual customers) in the software organizations. However, the Pearson chi-square statistic of association requested for testing the premise that type of product business and SPI success are independent of each other was not significant. Thus, there was no association between type of product business and SPI success.

Finally, we computed the Pearson chi-square statistic for testing the possible independence of having a quality system in use and SPI success. As shown in *Table 8.28*, the chi-square statistic was highly significant ($\chi^2 = 8.92$, $df = 1$, $p < 0.005$). Thus, we rejected the hypothesis of independence and concluded that there is an association between having a quality system in use and experiencing SPI success.

8.4.9 Exploring the effects of the respondents' characteristics

To examine the possible effects of the respondents’ characteristics on SPI success, we compared the successful and less successful organizations with respect to the respondents’ experience, education, and job function. Experience was measured along two dimensions: (1) company experience and (2) software experience. Company experience was defined as the number of years worked in the organization, while software experience was defined as the total number of years worked with software development. Thus, we compared the differences of the means between the successful and less successful organizations using two-tailed *t*-tests.

Results of these tests are summarized in *Table 8.29*. For each construct the table provides the mean score, standard deviation, and *t*-value.

The results showed no statistically significant difference between successful and less successful organizations with respect to software experience. However, respondents in successful organizations reported higher levels of company experience ($t = 1.81$, $p < 0.1$) than the respondents in the less successful organizations.

In addition to experience, we compared the respondents' level of education and their job function between the successful and less successful organizations using Pearson chi-square tests of association. Results of these tests are summarized in *Table 8.30*. For each characteristic the table provides the frequency, percent, and the Pearson chi-square statistic. None of the chi-square values was statistically significant. Thus, we accepted both hypotheses of independence and concluded that there is neither an association between education and SPI success nor between job function and SPI success.

Table 8.29: Successful versus less successful software organizations.

	Successful (n = 39)		**Less successful (n = 41)**		
Experience	**Mean**	**S.D.**	**Mean**	**S.D.**	***t*-value**
Company experience	9.56	7.21	6.98	5.49	1.81*
Software experience	12.55	8.26	10.07	7.87	1.37

* $p < 0.1$
1. All *t*-tests are two-tailed.

Table 8.30: Successful versus less successful software organizations.

	Successful (n = 39)		**Less successful (n = 41)**		
	Frequency	**Percent**	**Frequency**	**Percent**	χ^2
Education:					.30
Bachelor's degree	13	33.3	13	31.7	
Master's degree	24	61.5	26	63.4	
Doctoral degree	1	2.6	2	4.9	
Other	1	2.6	-	-	
Job function:					.83
Quality manager	11	28.2	8	19.5	
Software manager	28	71.8	33	80.5	

1. All chi-square tests are two-tailed.

8.5 Chapter Summary

In this chapter, we first reported on our experiences with goal-oriented measurement and participative software process assessment. The main lesson learned from these case studies was that participation combined with an explicit business orientation were the two most important factors for SPI success.

Subsequently, to further validate the learning cycle in our dynamic model of SPI, we reported on a separate empirical study on the utility of formal routines to transfer knowledge and experience. The results show that developers are rather skeptical at using written routines unless they participate in developing them, while quality and technical managers are taking the utility of routines for granted.

Then, we reported the results of the main, quantitative survey. We started by evaluating the analytical assumptions underlying the analyses used for testing the hypotheses by examining the statistical properties of the independent and dependent variables. We examined the assumptions of normality, homoscedasticity, linearity, and independence of the error terms, as well as any potentially influential observations (outliers) and the degree of collinearity and multicollinearity to validate the applicability of parametric statistics. The results of these tests showed that all variables were within the acceptable range for the normal distribution assumption, and that the assumptions of homoscedasticity, linearity, and independence of the error terms were supported. Furthermore, no influential observations were identified, and the investigations of collinearity and multicollinearity indicated no problems. In other words, *there were no extreme violations to the basic assumptions underlying the chosen data analysis techniques* that could justify the use of less powerful non-parametric statistics.

Next, we tested the hypotheses using bivariate correlational and partial correlational analyses as well as multiple regression and stepwise regression analyses. Results of the bivariate correlational analyses showed support for hypotheses 1 – 6 and results of the regression analyses showed support for hypothesis 7. However, the regression analyses did not support hypotheses 2 and 6 provided that all hypotheses are considered as one set. Considering both bivariate correlation analyses and multiple regression analyses, we find strong support for hypotheses 1, 3, 4, 5, and 7, and partial support for hypotheses 2 and 6 (see *Table 8.20*).

Finally, we examined the relationships between the success factors and the contextual variables. Specifically, we examined the effects of environmental conditions and organizational size on the success factors and on SPI success. The results showed that there are important differences between small and large organizations, specifically with respect to dealing with a turbulent environment. Also, we examined the effects of other organizational characteristics of the respondent's organizations on SPI success. These characteristics were industry sector, product business, and whether the organization had a quality system in use or not. To examine the possible effects of the respondents' characteristics on SPI success, we compared the successful and less successful organizations with respect to the respondents' experience, education, and job function.

In the next chapter, we discuss the results reported in this chapter and their implications for theory, methodology, and practice.

CHAPTER 9

Discussion and Implications

"The most fruitful developments have always emerged where two different kinds of thinking met."

– Werner Heisenberg

The goal of this study was to examine the importance of organizational issues to better understand how software organizations can enable SPI. We were interested in answering the following three questions: (1) What are the key learning processes in successful software organizations? (2) What are the key factors of success in software processes improvement? (3) What are the relationships between organizational context and modes of learning in software organizations?

The results from this study suggest that the dynamic model of SPI presented in *Chapter 5* is a useful conceptualization of process improvement in software organizations. The model answers the three research questions by explicitly distinguishing between three major sets of variables that influence outcome: learning cycle, facilitating factors, and organizational context. Based on this model, this study has qualitatively analyzed the learning cycle and quantitatively analyzed the impact of the facilitating factors on the outcome and the impact of organizational context on modes of learning. The major finding, that organizational issues are significant predictors of SPI success, emphasizes the importance of creating enabling contexts within software organizations. This has several important implications.

In this chapter, we discuss the answers to the research questions along with three sets of such implications. The first involves theoretical implications, offering new insights into a dynamic model of SPI that come out of the findings of this study. The second is related to methodology, and involves contributions regarding the conduct of SPI research. Finally, the third set involves practical implications and recommendations for software managers and software developers interested in implementing a successful SPI program within their organizations.

We use a *dialectic* approach to discuss the implications for theory and methodology. This approach accepts the diversity of assumptions and knowledge claims as an inevitable aspect of organizational learning and SPI, and it attempts to use the differences among competing perspectives as a means of constructing new modes of understanding. In contrast to approaches that try to eliminate diversity and uncertainty by searching for the "best practice", the dialectical approach deliberately counterposes the insights of different perspectives in the hope that a completely new mode of understanding will emerge from the debate generated by this opposition.

The importance of dialectic thinking in software development has also been noted by (e.g. Bjerknes, 1991; Dahlbom and Mathiassen, 1993, 1997; Sodan, 1998; Øgrim, 1993), and in the study of organizations by (e.g. Benson, 1983; Morgan, 1997; Tetenbaum, 1998). Furthermore, dialectic thinking is also seen in contemporary research that provides "trans-cultural" bridges between the "two cultures" of science and the humanities (Rosen, 1994).

9.1 The Learning Processes in SPI

The first research question focused on identifying the key processes that comprise the learning cycle of successful software organizations. A theoretical investigation, an in-depth pilot case study (Case A), a cross-case analysis of twelve software organizations (*Case B*, *C*, and *D*), and our own consulting experience provided the background for answering the question. Grounded in these data, we identified the foundations of the learning software organization (*Chapter 4*) and constructed a dynamic model of SPI (*Chapter 5*).

9.1.1 Foundations

How we perceive knowledge and the process of coming to know provides the basis for SPI practice. If we believe that software developers passively receive information, then priority will be on knowledge transmission and associated tools. If, on the other hand, we believe that they actively construct knowledge in their attempts to make sense of their world, then SPI will likely emphasize the development of meaning and understanding.

In accordance with the latter, the findings from this study, combined with our prior experience, led us to an emphasis on SPI as organizational learning – as a collective ability expressed in and through the organizations' product- and service-oriented practices related to software development. Furthermore, it led us to emphasize a more adaptive and active view of knowledge rather than the traditional static and passive view. As a result, we derived the following four foundations of the learning software organization:

- **Social learning**, which focuses on SPI as a social, collaborative activity within the context of a learning software organization;
- **Sensemaking**, which is aimed at constructing meaning and expressing the basic assumptions and values that are vital to the software organization and its members;
- **Knowledge creation**, which is aimed at generating new knowledge and new competencies that enable or broaden the software organization's potential range of actions;

- **Purposeful action**, which focuses on using the new interpretations and new knowledge to construct improved courses of action.

Accordingly, we found that successful software organizations were able to enable sense-making, knowledge creation, and purposeful actions in order to learn and improve their software processes (see *Chapters 4* and *5*). This suggests that successful SPI requires a commitment to learning and that organizational knowledge is constructed through the collaborative action of organizational members. This view on SPI corresponds to a design perspective on software (Adler and Winograd, 1992; Winograd, 1996; Winograd and Flores, 1986) and knowledge management (Brown and Duguid, 2000a, b; Fischer and Ostwald, 2001). In this perspective, working, learning, and SPI is related in much the same way as Brown and Duguid's (1991) unified view of working, learning, and innovation.

Based on these insights, we constructed a dynamic model of SPI, which embraces the learning processes and the key factors for success required for describing software organizations as learning systems. A major concern was to establish a conceptual framework, which structured the organizational learning process, thus facilitating the diffusion of knowledge and experience within and between groups of software developers.

Furthermore, we found that organizational learning and SPI is a continuous and simultaneous dialectic interplay between two components: the knowledge that the organization has established over time, and the knowing of the organization's members in their respective contexts. We will discuss each of these in turn.

9.1.2 Communities of practice

Throughout this study we have focused on local knowing as an ongoing process of social construction and collective action that is embedded in the software organization's tasks, relationships, and tools (see *Sections 4.4.4* and *5.3.1*). This conceptualization has turned out to be an accurate description of the reality experienced by software developers (e.g. *Case A*, *B*, and *C*); within the context of SPI they are reflective practitioners who struggle to understand and solve ill-defined problems.

We found that *learning is a fundamental part of software development, which could not be divorced from the specific context in which it takes place*. This observation supports Brown and Duguid's (1991) perspective on learning-in-working and that lessons-learned cannot easily be transferred from one setting to another. Therefore, both expertise and the coordination of such expertise in a specific context are important for improving the performance of a software team (Faraj and Sproull, 2000). Consequently, any routines, generalizations, or other means that strip away context should be treated with caution. This suggests that software problems are not given but must be framed and solved in each particular case. Furthermore, it suggests that software developers are the prime knowledge creators in software organizations, and that knowledge creation is an intrinsic part of practice.

Lave and Wenger (1991) explained this sort of simultaneous working and learning in terms of both the practice and the community. They argued that learning a practice involves becoming a member of a "community of practice" and thereby understanding its work and its language from the inside. This perspective has important implications for SPI, since learning cannot simply be viewed as a matter of acquiring information; it requires developing the "disposition", "demeanor", and "outlook" of the practitioner (Brown and Duguid, 2000b).

Furthermore, like Orr's (1996) study of technical reps and Wenger's (1998) study of claims processors, our findings revealed the importance of the group to both *what* and *how* software organizations learn.

Thus, the practice of the software team turned out to be the most important environment for SPI since it both shapes and enables organizational learning. This implies that *SPI is something that cannot be isolated from the ongoing practice of software development*, which, in turn, has important consequences for the ways in which SPI activities should be organized. An important challenge, therefore, is the need to acquire a new mindset – a mindset that no longer views software developers as passive receivers of information, but as *active constructors* of knowledge. Knowledge is not something that is handed down from above; it is constructed collaboratively in the context of work.

Such a practice and community orientation has also been noted in other studies of software development. For example, *Communications of the ACM* had a special edition devoted to the topic in its October 1993 issue. In that issue, the lead article by Constantine (1993) introduced a concise theoretical framework for making sense of the diverse possibilities in the organizing of software development as a collective human activity.

In their study of a software design team, Walz *et al*. (1993) examined the role and process of knowledge acquisition, sharing, and integration. Their study supports our observations of the importance of context-sensitive learning and that much of the knowledge that is part of a team's memory is not captured by formal documents (see *Section 8.3*). Furthermore, studies by Faraj and Sproull (2000), Glass (1999), Guinan *et al*. (1998), and Sawyer and Guinan (1998) have shown that social processes, such as the level of informal coordination, are more important in explaining the variance in software team performance than software methods and tool usage. Also, Curtis *et al*. (1988), DeMarco and Lister (1999), and Scacchi (1984) emphasized the importance of understanding how human and organizational factors affect the execution of software development tasks and, consequently, that the development and improvement of software systems must be analyzed as a *behavioral* process.

Since knowing is not defined outside the local "regime of competence" (Wenger, 1998), software teams create boundaries, or discontinuities, over time between those who have been participating and those who have not. So, while the creation of local knowledge deepens the insights of the team, it can simultaneously create problems for organizational communication, coordination, and collaboration.

However, boundaries are also places where new and innovative practices often start. This suggests that organizational learning require a balance between local practice and organizational reification. On the one hand, software teams are organizational assets through the depth of engagement and learning they develop. On the other hand, the locality of such learning entail the danger that useful connections beyond the boundaries of any given practice may not be apparent or sought. Thus, the localities and boundaries created by software teams reflect and shape the learning processes in SPI.

This suggests that the learning architecture of a software organization be composed of both communities and boundaries. "Brokering", i.e. "connections provided by people who can introduce elements of one practice into another" (Wenger, 1998, p. 105), is an important type of connection for bridging local practices. For sustained organizational learning and SPI, however, brokering needs to be supplemented by another type of connections: "boundary objects". In the next section, therefore, we discuss the role of organizational memory in providing such connections between local practices.

9.1.3 *Living organizational memory*

A central component of a software organization's learning architecture is a repository for storing knowledge – an organizational memory. However, as we saw in *Case C*, the mere presence of an organizational memory system does not ensure that software organizations will learn. For sustained organizational learning and SPI, two seemingly disparate goals must be served simultaneously. First, organizational memories must serve work by making stored information relevant to the situation by helping software developers make sense of the problems they face. Second, organizational memories must be extended and updated as they are used to support work practices. This suggests that organizational memories are dynamic and collaborative information spaces, which are actively integrated into the software processes and social practices of the team that constructs them.

While the problems facing software teams are unique, they are often similar to previously solved problems. This involves both problems and opportunities since the process of interpreting organizational knowledge must deal with prior experience and an already internalized body of knowledge – it cannot construct local reality *ex nihilo*. This presents a problem to SPI because the already internalized knowledge has a tendency to persist. Whatever new knowledge is to be internalized must somehow be superimposed upon the already existing knowledge. There is, therefore, a potential problem of consistency between the original and the new internalizations. To establish and maintain such consistency presupposes conceptual procedures to integrate different bodies of knowledge.

Traditionally, such integration has been the realm of knowledge engineers, members of SEPG's, or the analysts in Experience Factories. In a social construction perspective knowledge integration is part of the software developers' daily work. This presents a new challenge, since most developers do not consider this as part of their core responsibility. A major concern for our approach to SPI, therefore, is to capture the knowledge generated from local work processes without extra effort by the developers, and to help them consolidate and globalize this knowledge in the form of experience packages that can be stored in organizational memory.

However, a general observation is that many such "experience bases", which are filled up with presumably useful lessons learned, tips, and data, are not being used. There could be several explanations for this. One is that many such experience bases are created by knowledge engineers or quality engineers and filled with what managers think is useful for the people they manage. But as we have seen (e.g. *Case C*), the differences between occupational cultures are often so large that it should come as no surprise that developers usually don't find the managers' tips very useful. Interestingly, experience bases that are filled with individual developer's ideas are not of much help either. In fact, "The more a database contains everyone's favorite idea, the more unusable it becomes." (Brown and Duguid, 2000a, p. 79).

An example of getting past these problems is the way in which Xerox designed their worldwide Eureka database to preserve resourceful ideas over time and deliver them over space (Brown and Duguid, 2000a). Just as knowledge is circulated within the scientific community, Xerox established a process of peer reviews, in which acknowledged experts review any tips or ideas prior to inclusion in the database. Consequently, the reps using the system know that the tips that have survived the review process – and the database as a whole – are relevant, reliable, and probably not redundant. Furthermore, and also in accordance with the traditions of the scientific community, the reps don't get paid for their tips. However, their names are attached to the tips so that those who submit good tips earn positive recognition, thus, building social capital through the quality of their tips.

Furthermore, in our constructivist approach to SPI, the information needs of individual developers or software teams are unpredictable. Their needs for information arise from a particular situation in which developers struggle to understand a problem. This situation dictates the information demand and provides the context for learning. Such on-demand information, thus, integrates working and learning, because the need for learning comes from work, and because the learning takes place within the context of the work situation. This is similar to the unified view on working, learning, and innovation expressed by Brown and Duguid (1991).

Also, we have seen that the informality found *within* the local practice of software teams differs from the explicitness and formality often demanded *between* teams (e.g. *Case A* and *C*). This has important implications for the ways in which technology can support organizational memories. Basically, such technologies must be able to support different degrees of formality depending on the types of processes they are meant to support. This means that *technologies for organizational memory systems must not only support the diffusion of existing knowledge, they must also support the creation of new knowledge*.

Thus, supported by the views of Fischer and Ostwald (2001), the preceding discussion suggests that software organizations should construct living organizational memories with due consideration to the following design principles:

- They are information spaces owned by the people and communities who use them to develop software, not by management or the quality department.
- They support the collaborative and evolutionary design of complex software systems by providing a means to integrate the many contributions of many people.
- They are open and evolvable systems, serving not only as information repositories but also as mediums of communication, coordination, collaboration, and innovation.
- They can evolve through many small contributions by many people rather than through large contributions by a few people (as has been the case for previous knowledge-based systems).

9.2 *The Key Factors of Success in SPI*

In addressing the second research question we performed a quantitative survey to investigate the key factors of success in SPI (see *Section 8.4*). The results of the bivariate correlational and multiple regression analyses indicate support for all of the hypotheses in the proposed theory and demonstrate the existence of important factors for SPI success (see *Table 8.20*). These findings add an important new dimension to ESE research in that they verify the importance of organizational factors for SPI success. This suggests that, rather than trying to imitate technical procedures, software organizations should focus their SPI efforts on creating an organizational culture within which these procedures can thrive. This differs substantially from that found in most of the existing SPI literature, which focuses almost entirely on software engineering tools and techniques.

Our analyses showed that SPI success depends critically on six organizational factors: business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge. In the following subsections we discuss the importance of each of these factors.

9.2.1 Business orientation

Business orientation, that is, the extent to which SPI goals and actions are aligned with explicit and implicit business goals and strategies, was identified as one of the factors with the strongest influence on SPI success ($r = 0.62$, $p < 0.0005$; $pr = 0.61$, $p < 0.0005$). This supports the hypothesis that SPI success is positively associated with business orientation.

This finding is important, and suggests that both *practitioners and researchers should direct significant effort toward understanding shared domain knowledge between software and business executives*. However, as we pointed out in *Chapter 5*, there are two basic conditions for establishing such connections and making communications between these groups effective. First, each group must respect the expertise of the other, and must acknowledge the relevance of that expertise to their own problems. Second, each group must have sufficient knowledge and understanding of the other groups' problems to be able to communicate effectively about them.

It is important to remember that experience has shown that such shared understandings are unlikely to occur unless a sufficient number of members of each group have had actual experience with the activities and responsibilities of the other groups (Simon, 1991). This suggests that the software organization should be active in creating the possibilities for such connections to be formed, for example through personnel rotation programs, and promotion and recruitment strategies.

The critical importance of a business orientation in SPI suggests that software organizations should focus on both short-term and long-term alignment of SPI goals with business goals. They should focus attention on how to both achieve a high degree of mutual understanding of current objectives as well as on how to achieve congruence of long-term SPI goals with business strategies. The importance of business orientation was also confirmed by our studies in *Case A* of goal-oriented measurement in Nera AS (see *Section 8.1*) and by our studies in *Case B* of software process assessment in four of the SPIQ companies (see *Section 8.2*). One of the key lessons learned from the latter study was that the assessment should be closely aligned with and tailored to the company's overall strategy. Without this it would be hard to get acceptance by the managers, which would lead to fewer resources and low priority.

Thus, the importance of aligning SPI goals with business goals implies that if the SPI strategy is not kept in line with changes in the business strategy, there is a risk that the organization's software processes may end up as a burden rather than advance the business. Therefore, our results also suggest that understanding the business and organizational context is critical for achieving alignment between SPI activities and business strategy, which in turn is of paramount importance for the success of the SPI program. Similarly, by viewing the organization as a living organism, Zahran (1998) compared the introduction of a new process with transplanting a new organ: If the new process is not aligned with the business strategy or does not match the organization's culture, it will be rejected by the "organizational" body.

9.2.2 *Involved leadership*

Involved leadership was defined as the extent to which leaders at all levels in the organization are genuinely committed to and actively participate in SPI. It had a strong and highly significant correlation with overall SPI success ($r = 0.55, p < 0.0005$; $pr = 0.54, p < 0.0005$), which supports the hypothesis that SPI success is positively associated with involved leadership.

A surprising result, however, is the insignificant importance of involved leadership in predicting SPI success. All the main studies on which this investigation is based (e.g. Grady, 1997; Humphrey, 1989, 1997; Zahran, 1998) share a strong belief in the importance of management commitment for the successful implementation of SPI. Furthermore, the quality management literature (e.g. Crosby, 1996; Deming, 1986; Juran and Godfrey, 1999) and the organizational learning literature (e.g. Argyris and Schön, 1996; Nonaka and Tacheuchi, 1995; Senge, 1990) also seem to share a strong belief in the importance of management commitment for improving organizational performance. With this background, it is surprising, both from a theoretical and practical perspective, that involved leadership, defined as the extent to which leaders at all levels in the organization are genuinely committed to and actively participate in SPI, is a non-significant predictor of SPI success. There could be several explanations for this.

First, as we found in *Section 8.4.1*, multicollinearity should not have a substantial impact on the estimated regression variate in this investigation. However, the results of the multiple regression analysis show that multicollinearity, nevertheless, does have an impact on the composition of the variate, and that correlations among the independent variables may make some variables redundant in the predictive effort. This is the case for involved leadership and also for exploration of new knowledge. However, as shown in *Section 8.4.2*, this does not reflect their individual relationships with the dependent variable. Instead, it indicates that in a multivariate context, they are not needed together with the remaining set of four independent variables (participation, business orientation, measurement, and exploitation) to explain the variance in SPI success. Therefore, we cannot determine the importance of involved leadership based solely on the derived variate, since relationships with the other independent variables may "mask" relationships that are not needed for predictive purposes, but represent substantive explanatory findings nonetheless.

Second, the role of management is often to ensure that SPI goals and actions are closely aligned with business goals and strategies, which is indicated by the highly significant ($p < 0.0005$) correlation coefficient of 0.63 in *Table 8.11* between business orientation and involved leadership. This suggests that involved leadership is important through business orientation. The following statement from one of the software managers participating in the management experience forum in SPIQ substantiates this:

> Software managers can do whatever they want, as long as they operate within the organization's overall plans and strategies. Therefore, management commitment is most important in guiding SPI activities in the right direction.

This is also in agreement with Grady's (1997) findings from his longitudinal studies in Hewlett-Packard that business aspects in the organization's SPI efforts has a strong influence on management commitment. Furthermore, Debou and Kuntzmann-Combelles (2000) argued that management commitment implies business orientation and vice versa.

Finally, a large part of the published, and thus reviewed, studies on quality management, organizational learning, and SPI are from the United States. There are, however, important cultural differences between the United States and Europe in general (Messnarz, 1999b), and between the United States and Scandinavia in particular (Ehn, 1992). The Norwegian Work Environment Act, for example, stipulates that the employer has to negotiate with the employees before making "major changes" in production. The relatively high degree of workplace democracy and socio-technical tradition in Scandinavian countries suggests a higher importance of employee participation and a correspondingly minor importance of involved leadership in predicting SPI success. This is also indicated by the results of the multiple regression analysis that showed that together with business orientation ($\beta = 0.245$, $t = 2.78$, $p < 0.01$), employee participation ($\beta = 0.297$, $t = 3.80$, $p < 0.0005$) was the most important factor in predicting SPI success.

9.2.3 *Employee participation*

Together with business orientation, *employee participation*, i.e. the extent to which employees use their knowledge and experience to decide, act, and take responsibility for SPI, was identified as one of the factors with the strongest influence on SPI success ($r = 0.55$, $p < 0.0005$; $pr = 0.59$, $p < 0.0005$). This supports the hypothesis that SPI success is positively associated with employee participation. This is not surprising, since employee participation and the way people are treated, has been noted to be a crucial factor in organizational management and development ever since the famous productivity studies at Western Electric's Hawthorne plant in the 1920s (Mayo, 1933, 1945). The results of these studies started a revolution in management thinking, showing that even routine jobs can be improved if the workers are treated with respect.

Also, the interviews conducted in *Case C* to explore the diffusion of knowledge and experience in software organizations (see *Section 8.3*) strongly supports the importance of participation as a key factor for success. The results of these interviews showed the importance of participation during the development and introduction of formal routines. Viewed in the light of our results, it is not surprising that research on formalization often presents conflicting empirical findings regarding its efficiency. This divergence can be explained by the fact that most prior research has focused on different *degrees* of formalization, and has paid insufficient attention to different *types* of formalization.

Adler and Borys (1996) expanded on this in their study of the enabling and coercive types of formalization. In an *enabling* type of formalization, procedures provide organizational memory as a resource to capture lessons-learned or best practices. The core idea is that "good" procedures are those seen as valuable resources that help practitioners meet customers' needs. By contrast, in a *coercive* type of formalization, procedures are a substitute for, rather than a complement to, commitment. Instead of providing committed employees with access to accumulated organizational learning and "best practice" templates, coercive procedures are designed to force reluctant compliance and to remove noncompliant efforts. View in this light, formal routines are perceived as a considerable barrier to organizational learning and SPI.

Our results regarding the developers' assessment of the routines in their respective organizations closely resemble the coercive type of formalization. However, the developers were clearly not against formal routines. In fact, they expressed views in favor of such routines, especially those that captured prior project experience. Contrary to the existing routines, which they deemed coercive, they sought routines of the enabling type. Furthermore, our study in *Case B* of tailor made assessments to focus software process improvement activities in four of the SPIQ companies, showed that a structured process emphasizing participation in each and every step was a key factor for success (see *Section 8.2*).

This suggests that *people tend to support what they have participated in creating*, or to use Berger and Luckmann's (1966) words: "it is more likely that one will deviate from programmes set up for one by others than from programmes that one has helped establish oneself." (*ibid.*, p. 80). Thus, we learn a great deal more from our own experience than we do from those who are experienced.

9.2.4 Concern for measurement

Concern for measurement was defined as the extent to which the software organization collects and utilizes quality data to guide and assess the effects of SPI activities. It had a strong and highly significant correlation with overall SPI success ($r = 0.50$, $p < 0.0005$; $pr = 0.48$, $p < 0.0005$), which supports the hypothesis that SPI success is positively associated with concern for measurement. This suggests that *significant progress in software development depends on an ability to measure aspects of the process and the resulting artifacts, and to make analytical judgements based on the results of these measurements*. Thus, a lack of measurement leads to a lack of empirical validation of software techniques and ultimately to an inability of the organization to evaluate new (or old) ideas and to separate sound and useful practices from current fads.

Traditionally, measurement systems are used to provide upper management with data for decision making, and to assess the effects of organizational actions. Although this is still important, our concept of measurement not only includes these traditional uses; it also includes availability and feedback of data, and the use of data to guide SPI actions. So, while measurement in itself can have a significant effect on organizational behavior (Nadler, 1977), *Case A* suggest that the most effective use of data for organizational learning and SPI is to feed the data back in some form to the organization's members (see *Section 8.1* and Dybå, 2000e). Such feedback regarding actual performance not only motivates change in the behavior of individuals, groups, and organizations; it can also guide change in a specific direction. However, in order for feedback to change behavior through these mechanisms the data must be perceived as valid and accurate. Also, the conditions surrounding the feedback process must support the non-threatening use of data for identifying and solving problems.

This suggests that without clear and consistent feedback to help monitor software teams' progress toward their goals and the effects of their actions, it is difficult to learn. Besides, our results demonstrate that measurement is meaningless without interpretation and judgement by those who will make decisions and take actions based on them. Within the context of SPI, therefore, *it is important that measurement systems are designed by software developers for learning, rather than by management for control*. Also, as we saw in *Case A*, an important observation is that a few measures that are directly related to the software process are better than a multitude of measures that produce a lack of focus and confusion about what is important and what is not (see *Section 8.1*). Moreover, such measures should be closely aligned with the organization's business strategy (Rifkin, 2001).

Our findings thus suggest that data based methods in general and feedback in particular are effective tools for the successful application of SPI. Furthermore, our findings support Nadler's (1977) conclusions, suggesting that the process of using data is an important factor influencing how effective measurement systems will be, that feedback is most effective when combined with different interventions, and that different situations require different feedback designs. Therefore, knowledge about *how things are* can be a potential force that moves software organizations toward *how things should be*.

9.2.5 Learning strategy

Two fundamentally different learning strategies were identified as part of our investigation: *exploitation* of existing knowledge and *exploration* of new knowledge. Consequently, we defined learning strategy as the extent to which a software organization is engaged in the exploitation of existing knowledge and in the exploration of new knowledge. Both learning strategies had a significant correlation with overall SPI success, which supports the hypotheses that SPI success is positively associated with the exploitation of existing knowledge ($r = 0.59$, $p < 0.0005$; $pr = 0.59$, $p < 0.0005$) and with the exploration of new knowledge ($r = 0.21$, $p < 0.05$; $pr = 0.25$, $p < 0.005$).

However, as in the case of involved leadership, exploration of new knowledge had an insignificant importance in *predicting* SPI success. We see two plausible explanations for this. First, as we explained with respect to involved leadership, relationships with the other independent variables may "mask" relationships that are not needed for predictive purposes, but represent substantive explanatory findings nonetheless. Therefore, we cannot determine the importance of exploration of new knowledge based solely on the derived variate.

Second, our concept of exploration includes issues such as innovation and creativity, questioning of "established" truths, flexibility, minimum critical specification, and diversity, which, within a socio-technical perspective, can be seen as closely related to conceptions of participation. This is indicated by the highly significant ($p < 0.0005$) correlation coefficient of 0.46 in *Table 8.11* between employee participation and exploration of new knowledge, which suggests that exploration of new knowledge is important through employee participation.

Thus, the results suggest that both modes of learning are important for successful SPI. More importantly, they suggest that *software organizations should balance their learning efforts between exploitation and exploration* (see *Section 8.4.7*). Thus, relying on past experience and data as the sole basis for process improvement necessarily leads to an improvement strategy that is past oriented, supporting and legitimizing past and current practices. Consequently, such strategies may lead to a lack of novelty and fresh perspectives as well as an over-emphasis on the testing of already established theories and ideas, or as one of the software managers in our study expressed it:

> Relying on past experience alone to prepare for the future is like steering a car by looking in the rear mirror. In stable and well-known territories it might do the job. In new and unknown territories it will most certainly lead to disaster.

In our experience, the situation is even more dynamic, it is more like walking across a glacier in which there are some safe routes, some unsafe routes and some routes you don't yet know if they are safe or not. However, ice flows – it moves constantly. Therefore, routes that were safe yesterday are not necessarily safe today and vice versa.

9.3 *The Effects of Organizational Context*

The third research question focused on the relationships between organizational context and modes of learning in software organizations. As we saw in the previous section, two modes of learning were identified in this study: exploitation of existing knowledge and exploration of new knowledge. Also, as we saw in *Section 5.2*, we identified two contextual variables to capture the most influential sources of variation: environmental turbulence and organizational size. Effects of *organizational size* were compared by contrasting large organizations with small organizations, while the effects of *environmental turbulence* were examined by contrasting stable environments with turbulent environments.

9.3.1 Environmental turbulence

As reported in the previous chapter (see *Section 8.4.5*), no statistically significant differences were found between organizations operating in stable or turbulent environments for the dependent and independent variables. Furthermore, there was no statistically significant difference, neither in the mean level of environmental turbulence between large and small organizations nor between successful and less successful organizations.

This finding is in agreement with e.g. Deephouse *et al.*'s (1996) results from a survey of 87 projects from different organizations regarding the effectiveness of software processes on project performance. Along with such practices as process training, user contact, design reviews, and prototyping, they found that stable environments had little impact on project outcomes.

Accordingly, there is no support for the commonly held belief among both researchers and managers within the software community that success depends upon stability and the ability to predict changes in the environment. Predictability is a property of simple systems. Reality is different; the environment is not a simple system. On the contrary, it is highly complex, is not predictable, is not entirely knowable, and is definitely not controllable by the software manager or the software organization. The sooner we admit this to ourselves, the sooner we can develop more useful models for improving the process of software development.

So, while Humphrey (1989) and others (e.g. Florac and Carleton, 1999) have argued that stability and predictability lies at the heart of all process management and process improvement efforts, our results clearly show that this is not the only road to success. Indeed, our results suggest that such conditions have no influence on the level of SPI success.

This suggests that it's time we distance ourselves from the assumptions underlying the rationalistic, linear model of software engineering, and admit that reality for most software organizations is a non-deterministic, multi-directional flux that involves constant negotiation and renegotiation among and between the social groups shaping the software.

This requires an improvement approach that recognizes (1) the need for a dramatically shorter time-frame between planning and action; (2) that the planning of an action does not provide all the details of its implementation; and finally (3) that creativity is necessary to make sense of the environment. We consider *improvisation* as such an improvement approach to better understand the relationship between action and learning in software organizations. Furthermore, improvisation at the local level is not only a legitimate alternative to models of planned change, but also necessary for effective knowledge creation and SPI (Dybå, 2000b).

It is important to emphasize that *we must differentiate between environmental turbulence and organizational turbulence*. So, while there was no statistically significant difference in the mean level of environmental turbulence between successful and less successful organizations, respondents in successful organizations reported higher levels of company experience (t = 1.81, $p < 0.1$) than the respondents in the less successful organizations. Thus, since much of an organization's memory is stored in human heads, and only a little of it is put down on paper or held in computer memories, *turnover of personnel is a great threat to long-term organizational memory and successful SPI* (see *Section 5.3.3*).

9.3.2 *Organizational size*

The findings from this study showed that there are fundamental differences between small and large software organizations. With respect to organizational performance in general, large software organizations reported a higher level of overall SPI success (t = 2.18, $p < 0.05$) than small organizations. This is consistent with e.g. Terziovski and Samson's (2000) finding that larger companies tend to gain greater benefits from TQM than smaller firms. This is also as expected, given that an organization will grow larger with repeated success. However, when comparing small successful organizations with large successful organizations, there was no difference in the level of SPI success.

What's more important, however, is that large successful and small successful organizations differ fundamentally in their respective approach to SPI, specifically with respect to participation and the preferred mode of learning. As can be seen from *Table 8.27*, small successful organizations reported higher levels of employee participation (t = -2.44, $p < 0.05$) and exploration of new knowledge (t = -3.50, $p < 0.001$) than the larger organizations. Furthermore, while organizations with a quality system in use showed higher levels of SPI success than those who had no such system, industry sector and product business had no significant effect on success. Also, as we saw in the previous section, company experience was an important factor in differentiating successful from the less successful organizations.

Thus, contrary to the claims put forward in the quality management literature that ideal quality management should not be affected by contextual variables (see *Section 5.2*), these findings suggests that it does, indeed, when it comes to implementation issues. This is consistent with recent research on organization size (e.g. Groth, 1999; Lawler, 1997) and on quality management comparing large and small firms. For example, in a survey of 488 U.S. and Canadian firms, Ahire and Golhar (1996) found that, while small successful firms implemented TQM elements at least as effectively as large firms, small firms relied on an implementation strategy that involved significantly higher levels of employee involvement and flexibility than the larger firms' strategy.

Also, in a study of BPR in 134 Canadian enterprises, Raymond *et al*. (1998) found that the small and medium-sized enterprises (SMEs) perceived an overall level of advantages equal to that of the large enterprises. Moreover, they found that the SMEs approach to BPR was less rigorous, more resembling a socio-technical approach.

Our findings regarding the differences between large and small companies are also consistent with the views expressed by Mohamed Fayad's, Mauri Laitinen's, and Robert Ward's "Thinking Objectively" column in *Communications of the ACM* (1997-2001) regarding software engineering and SPI in small organizations. Similar views were also presented in the special section of the September/October 2000 issue of *IEEE Software* on software engineering in the small (e.g. Dybå, 2000b; Laitinen *et al*., 2000), and also by Brodman and Johnson (1994) and Batista and de Figueiredo (2000) regarding the implementation of CMM in small organizations.

Thus, to answer the third research question, the effects of organizational size and environmental conditions on the learning modes were examined by comparing the estimated marginal means of exploitation of existing knowledge and exploration of new knowledge, using two-way analysis of variance (ANOVA) with organizational size and environment as the two classification variables (see *Sections 8.4.5-8.4.7*).

It is interesting to note that, as there was no difference in the level of exploitation between small and large organizations regardless of the environment, there was a marked difference in the level of exploration (see *Figure 8.5b*). The results showed that small software organizations engaged in significantly ($p < 0.001$) more exploration in turbulent environments than large software organization.

This suggests that *the main difference between small and large software organizations is the ways in which they react to unstable and changing stimulus situations*. In small organizations, the response to a changed situation was an increased level of exploration. This supports the assertion that small software organizations in turbulent environments require learning strategies that are more closely aligned with explorative behavior, while at the same time promoting the exploitation of past experiences. As discussed in *Chapter 4* and also in (Dybå, 2000b), this is at the heart of an SPI strategy based on *improvisation*.

However, while most software managers agreed that changes in their competitive environment are fast and increasingly unpredictable, managers of large software organizations still relied on learning from experience to prepare for the future rather than exploring new possibilities. They tended to generate the same response even when the stimuli had changed. They kept doing what they did well, rather than risk failure. This behavior is supported by the organizational literature, which suggests that large organizations are less likely to change in response to environmental changes than small organizations. There could be many reasons for this mode of learning. Tushman and Romanelli (1985), for example, argued that increased size leads to increased complexity, increased convergence, and thus, increased inertia.

Likewise, Mintzberg (1989) postulated that the larger an organization is, the more formalized is its behavior. So, while small organizations can remain organic, large organizations develop bureaucracies with job specialization and sharp divisions of labor, emphasizing stability, order, and control. As a consequence, they often have great difficulties in adapting to changing circumstances because they are designed to achieve predetermined goals – they are not designed for innovation.

From a learning perspective, however, inertia develops as a result of the organization's performance history (Lant and Mezias, 1992). Large organizations tend to be successful since an organization will grow larger with repeated success. However, since success reduces the probability of change in a target-oriented organization (Cyert and March, 1963, 1992), large software organizations will be less likely to change when the environment changes.

Another explanation, which is also a direct consequence of using "best practice" models such as the CMM, is the institutionalized routines. In stable situations, such routinization can become an effective way of developing software, but it can also drive out the exploration of new alternative routines. The deeper these routines are grounded in the organizational culture, the more difficult they are to change and more easily they turn into an obstacle to improvement.

A further explanation is that the rationalistic approach to software development considers failure as unacceptable. This is consistent with the goal of promoting stability and short-term performance, as is the case in exploitation. In this situation, success provides an excellent foundation for increased reliability. On the other hand, success tends to encourage the maintenance of the status quo by generating single-loop learning. Failure, however, can generate double-loop learning, in which people question the assumptions behind their failures as well as the failures themselves. The absence of failure experiences can, therefore, result in decreased organizational competence when faced with changing and turbulent environments.

Therefore, it is important to note that *successful SPI requires tolerance for failure*, and that failure is an essential prerequisite both for learning and for challenging the status quo. We must, thus, be able to turn unexpected problems and failure into learning opportunities (Abdel-Hamid and Madnick, 1990).

9.4 Implications for Theory

The key to our dynamic model of SPI lies in understanding the dialectic nature of the learning processes and their enabling conditions (see *Chapter 5*). Most importantly is the recognition of the dialectical process between *generation* of new knowledge, through action, resulting in *organizational knowledge*; and the *interpretation* of this organizational knowledge, through sensemaking, into *local knowing*.

We have come up with several dialectic relationships in this study. However, our findings suggest that the following four relationships are specifically important for advancing the theoretical basis of SPI:

- The tacit – explicit nexus
- The local – global nexus
- The knowledge – action nexus
- The exploration – exploitation nexus

We integrate each of these dialectic relationships to form a dynamic synthesis. It is these syntheses that form the primary contributions of this research to the theory of SPI.

9.4.1 The tacit – explicit nexus

Our results (e.g. *Case C*) have given clear indications that *there is a deeply rooted tension between tacit and explicit knowledge at all levels in software organizations* (see *Section 8.3* and Dybå, 2000b). This is in line with observations made throughout the history of science, where we have witnessed a controversy regarding which types of knowledge that can be trusted (Popper, 1979). For example, within the university culture the notion of "objective" knowledge, which is deeply rooted in technical rationality, is given much more value than the experiential and practical skills embodied in social and working life activities.

However, to exclude the tacit (experiential) knowledge from the explicit (formal) knowledge is to undermine the diversity of social and cultural contexts of knowledge and the diversities of interaction. This leads to defining the potential of SPI mainly in terms of technological infrastructures, and defining the transfer and dissemination of knowledge in terms of formal routines and standards. The danger of this focus is that it is likely to exclude software communities and organizations whose knowledge does not fit into these engineering-oriented notions of the rationalistic tradition.

As we saw in *Chapter 5*, the learning cycle centers on the creation of both tacit and explicit knowledge, and more importantly, on the dynamic interchange between these two aspects of knowledge through generating and interpreting. Thus, a failure to build a dialogue between tacit and explicit knowledge can cause problems for organizational learning and SPI. For example, as we saw in *Case C*, purely relying on either formal routines or face-to-face interaction are two extremes that both have drawbacks (see *Section 8.3*). Neglecting a software team's need to make sense of organizational knowledge might easily lead to superficial interpretations, which has little relevance for local reality. On the other hand, as we discussed in *Section 9.1*, the shared practice created by pure socialization within a software team might be difficult to apply in fields beyond the specific context in which it was created.

Thus, *generating new explicit knowledge by articulating tacit knowledge and interpreting organizational knowledge to extend a software team's tacit knowledge base are the critical steps in the learning process*. The main reason for this is that both of these processes require personal commitment. Indeed, that is also in agreement with our finding that employee participation are one of the critical factors for success in SPI.

The inherent paradox in the tacit-explicit nexus is that while the explicit dimension makes knowledge and experience easier to diffuse and reproduce, at the same time it undermines the traditional forms of participation and collaboration at work. Thus bringing about a separation of the human work from the workplace. From a management point of view, externalization may give rise to new patterns of work and organization in which employees, customers, and products are seen as components of an information network. On the other hand, the separation of work from the workplace may also lead to a degradation of the traditional social features of the workplace culture, such as teamwork, participation, and trust. This is unfavorable in light of our findings that participation and company experience are important factors that differentiates successful from less successful organizations. *In balancing the tacit and explicit, therefore, the primary focus should be on the negotiation of meaning and usefulness rather than on the techniques of information acquisition and transmission*. While the latter techniques clearly need to be in place in the learning cycle, they should not become the primary focus of SPI.

Consequently, *the challenge facing SPI research is to establish a symbiosis between people and technology, and therefore, a symbiosis between the tacit and explicit dimensions of knowledge*. This symbiosis recognizes the essential contribution of the explicit knowledge as a global resource for knowledge transfer and development. However, it emphasizes that sustainable improvements depend upon local capacity for the interpretation, absorption, and use of knowledge in diverse situations, and this in turn depends on the level of interdependence between local and global knowledge. The tacit dimension of knowledge needs to be brought back into the organization's memory as a way of preparing individuals and teams to master the technology, rather than technology mastering them. Thus, to split tacit from explicit knowledge is to miss the point – the two are inseparably related.

9.4.2 The local – global nexus

Software development, like all human action, is always and at every moment confronted with specific conditions and choices. On the one hand, software developers select what they understand to be the *relevant aspects of organizational knowledge* and on the other hand, those *relevant aspects of the local conditions* within which their actions take place, and try to fit the two together. Moreover, software developers possess local knowledge that cannot be surveyed as a whole and, also, part of their knowledge originates from outside the organization. It is this tension between the organization specific *habitus* and the *local* conditions in which it is instantiated that explains why a software project's development process is neither a replication of an idealized process model nor, an *ex nihilo* construction.

Thus, in the context of our dynamic model of SPI, *the local and global are related levels of learning that always coexist and shape each other*. Therefore, we need to find new ways of interlocking local and global knowledge bases for software development, in order to meet the conditions for organizational learning and SPI.

Our findings from *Case C* showed that software developers prefer local knowledge that is close in space and time to global knowledge that is more distant. Executive managers and quality managers, on the other hand, prefer global knowledge (see *Section 8.3*). As can be seen from *Table 9.1*, local knowledge is primarily concerned with unique situations and actual processes, while global knowledge is concerned with abstract generalizations. Furthermore, by combining the local-global dimension with the tacit-explicit dimension, we see from *Figure 9.1* how we can categorize some of the main types of knowledge in SPI.

While local knowledge is effective for software teams and local management, it is difficult to generalize from one setting to another and, consequently, of limited use for others than its creators. However, these limitations to local knowledge are, as we have seen, in the areas that occupy executive managers and quality manager. So, while executive managers and quality managers tend to reward abstract conceptualizations and distancing from individual cases, software developers and local managers tend to reward concrete thinking and closeness to the individual case.

These differences can create a tension that can be detrimental for organizational learning and SPI unless we try to understand it and engage in efforts of building a symbiotic relationship between local diversity and global unity. In other words, *a major implication of this investigation is an increasing awareness and heightened recognition of the need to strike a balance between sensitiveness to unique, local contexts and the need to standardize uniform solutions across multiple contexts*.

Table 9.1: Characteristics of local and global knowledge.

Local Knowledge	**Global Knowledge**
• Concrete, qualitative descriptions of unique situations	• Abstract, quantitative descriptions of key performance indicators
• Representations of actual processes	• Representations of means and variances
• Connects outcome to the processes in the situation	• Representations of outcomes of abstract processes
• Close in space and time to practice	• Distant from practice

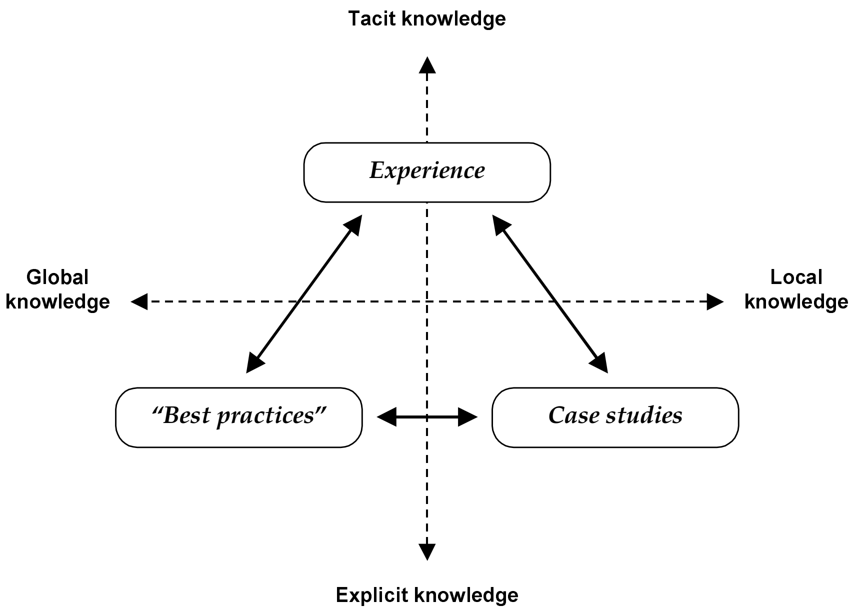


Figure 9.1: Different types of knowledge in SPI.

The problem with global knowledge and standardization in SPI is that uniformity produces context insensitive targets, methods, and processes. This view assumes a homogeneous understanding of organizational reality, separating knowledge from its context. This separation, then, contributes to widening the gulf between knowledge and experience (see *Section 9.5.3*). The risk of pursuing "best practice" approaches to SPI is, therefore, that it leads to the "one best way" of knowledge, imposing it as the "lowest common denominator" in which software developers and teams lose their contextual bearings.

However, *our findings suggest that improvement processes are highly contextually embedded* (see *Sections 8.1-8.3*) It must, therefore, be emphasized that the essential condition of knowledge transfer is that the local groups possess the educational, experiential, and learning infrastructures, which support the sharing, absorption, and use of global knowledge. Simple faith in the technical aspects of transferring knowledge and experience misses the essential condition of context and social mediation in making sense and purposeful usage of knowledge. All of a team's local knowledge, its ability to interpret data from a context and previous experience, its ability to discern the important from the unimportant, and to justify and make sense of action alternatives is impossible to embed or mimic in a global system.

As we have already seen, employee participation and business orientation are two of the most critical factor for success in SPI. With respect to the local-global nexus, participation represents the local, while business orientation represents the global. This suggests that within the norms of the organization's business strategy, participation should be intended to exploit the diversity within and between software teams to create a "space of possibility" (Gjersvik, 1993) that can be utilized in SPI. This way, the software organization can balance the local and the global by keeping its common structures and routines to a bare minimum, thus, avoiding unnecessary closure in its organizational learning and SPI activities.

9.4.3 The knowledge – action nexus

Being experienced, having knowledge, and balancing the tensions between the tacit and the explicit, and between the local and the global, is, as we have just discussed, of utmost importance to SPI. However, just as having knowledge about horses is not the same as an ability to ride, *to improve software development processes, knowledge, experience, and action are all mandatory*.

One of the insights from this investigation is that; while most of the software organizations in fact have knowledge of what to do in order to improve (see *Section 8.2*), they frequently fail in turning this knowledge into action. This might also explain why software education and general software experience did not influence the level of SPI success (see *Section 8.4.9*) – it is action that counts not theoretical knowledge alone. This is similar to the "performance paradox" observed by Cohen (1998): "Managers know what to do to improve performance, but actually ignore or act in contradiction to either their strongest instincts or to the data available to them." (*ibid.*, p. 30).

There are several possible explanations for this phenomenon. One is that the conception of knowledge as something explicit and quantifiable, leads organizations to overestimate the importance of the tangible, programmatic aspects of what competitors do, and underestimate the importance of the underlying philosophy that guides what they do and why they do it. There is, therefore, a danger that software organizations will adopt what Card (1991) called a "cargo-cult mentality" toward SPI:

> In the late 19th and early 20th centuries, rural natives of Melanesia, a group of islands in the southern Pacific, observed that valuable cargoes of food and other goods arrived by ship and plane at European harbors and airports. They reasoned that if they constructed harbors and airports (or at least things that looked like harbors and airports from a distance) valuable cargoes would arrive for them, too. So many rural Melanesians gave up their traditional activities of farming and fishing in the pursuit of cargo. Occasionally a ship or plane would arrive, but such occasions really were accidents (*ibid.*, p. 103).

Obviously, the Melanesians of that time only partially understood the system behind the movement of ships and planes. Card's (1991) example is, thus, an extreme example of what can happen when less knowledgeable people try to copy a successful behavior or practice without learning about the philosophy on which it is grounded.

Another reason for the observed gap between knowledge and action is that the most obvious solutions often tend to be discounted for the reason that they seem too ordinary. *The key lesson is, thus, to tackle obvious things*. Still another reason for the knowledge-action gap is that action inevitably involves a greater risk of failure than planning, decision-making, and talks. Furthermore, managers often believe that just because a decision has been made, something will happen. However, as Brunsson (1989) has shown in his analysis of organizational hypocrisy, talk, decision, and action is often not connected.

There are no easy answers to close the gap between knowledge and action. However, our findings suggest that too many managers and developers alike fall into the trap of the "cargo-cult mentality" wanting to learn "how" in terms of tools and techniques, rather than "why" in terms of underlying philosophy and guidance for action. Also, since learning by doing, by definition, eliminates the knowledge-action gap, this suggests that we should focus on theories that are grounded in industrial experience by developing practice-oriented theories.

Furthermore, evidence suggests that 70-80 percent of large-scale reengineering and quality improvement efforts do not produce the desired results (Cohen, 1998; Hammer and

Champy, 1993; Morgan, 1997). This suggests that rather than large-scale, complex, and abstract theories, we should develop actionable SPI theories that are close in space and time to practice, focusing on generating short-term, incremental gains and small wins, for example through the use of "microworlds" (Abdel-Hamid, 1996; Sauer *et al*., 2000; Senge, 1990). Moreover, such approaches should be improvisational in nature, valuing an attitude of action and the importance of learning by experimentation. To slightly rephrase Benjamin Disraeli, we should remember that: *action may not always bring improvement, but there is no improvement without action.*

9.4.4 The exploration – exploitation nexus

As we saw in *Section 9.2.5*, *a specific challenge for successful SPI involves balancing the refinement of the existing skill base with the experimentation of new ideas to find alternatives that improve on old ideas*. This insight is important and, unfortunately, it is not part of current "best practice" models. Therefore, to discuss the implications of the tensions between SPI strategies based on exploration and SPI strategies based on exploitation, we start by examining the implications of adhering to an exploitation strategy as devised by "best practice" models (see Dybå, 2000b).

Basically, there are three ways in which a software organization can improve its process capability using "best practice" models such as the CMM (Florac and Carleton, 1999): it can increase the average performance (i.e. the mean of the performance distribution), it can reduce the variance in performance (i.e. increase predictability), or it can use a combination of both. Increased average performance is a general feature of experiential learning, and it is clearly beneficial for competitive advantage. Reduced variance, however, is not necessarily an advantage. In fact, for most small software companies, *competition can turn reduced variance into a major disadvantage*.

To examine the "best practice" approach to improvement and its consequences for competitiveness, we consider a simple model devised by March (1991), which assumes that survival is based on comparative performance within a group of competing organizations. Furthermore, each single performance is drawn from a performance distribution specific to a particular organization. The mean of the distribution reflects the organization's ability level, and the variance reflects the organization's reliability.

For small software organizations, performance samples are also small. Relative position does not depend on ability alone but is a consequence of ability and reliability (Levinthal and March, 1993). Moreover, the competitive environment of most small software organizations is such that only the best survive – and survival depends on having an extreme performance draw. Thus, improving average ability helps relatively little and increasing reliability (reducing variance) can detrimentally affect survival.

In the extreme case where the organization faces only one competitor, increases in average performance always pays off, whereas changes in variance have no effect on competitive advantage. However, when there are several competitors, increases in either the mean or the variance have a positive effect. As the number of competitors increases, the contribution of the variance to competitive advantage increases. Ultimately, as the number of competitors goes to infinity, the mean performance becomes irrelevant (March, 1991).

The argument behind CMM's improvement approach is that as software processes are standardized and techniques are learned, the time required to accomplish development tasks will be reduced, productivity and the quality of task performance will be increased together with the reliability of task performance (Paulk *et al*., 1995). March's (1991) model implies that if the increase in reliability comes as a consequence of reducing the left-hand tail of the performance distribution (see *Figure 9.2a*), the likelihood of finishing last among several competitors is reduced, without changing the likelihood of finishing first. But, if process improvement reduces the right-hand tail of the distribution, it may easily decrease the chance of being best among the competitors despite increases in the organization's average performance.

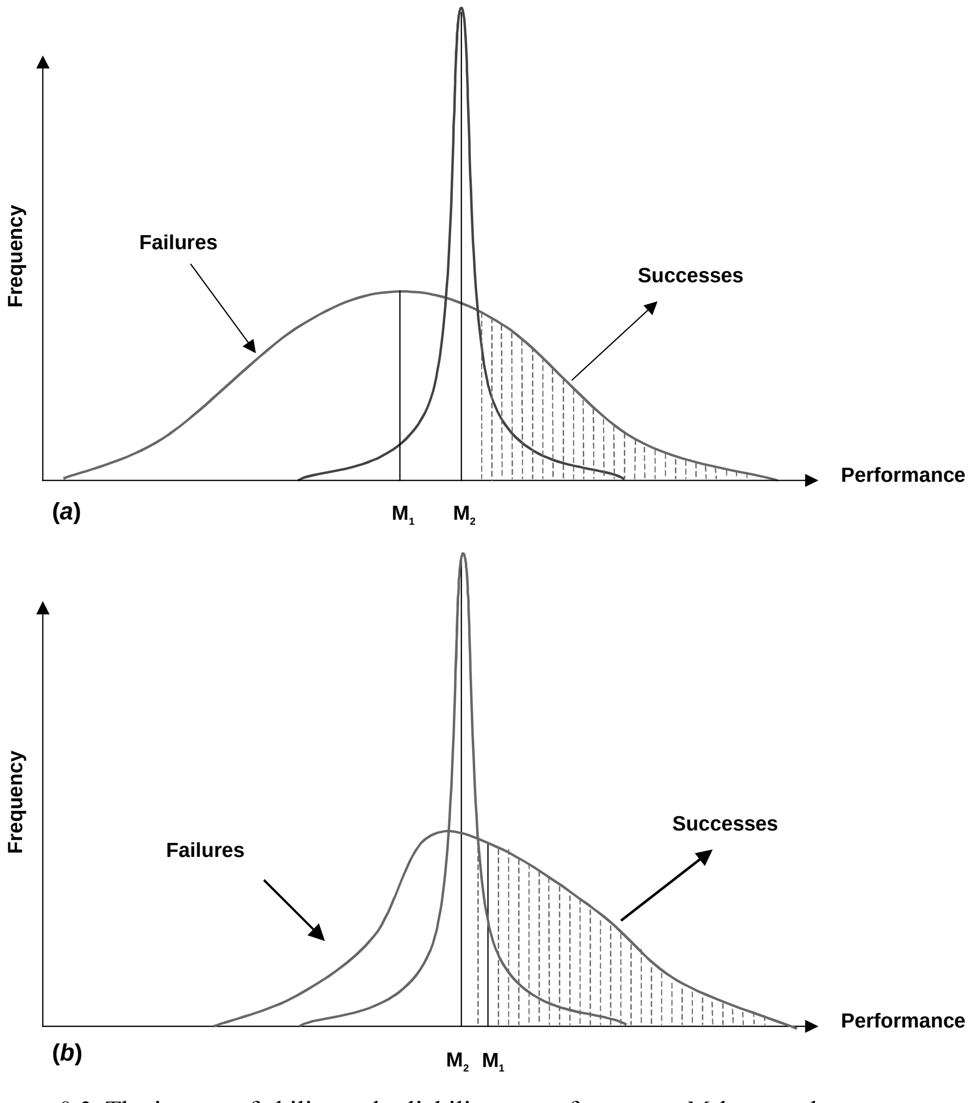


Figure 9.2: The impact of ability and reliability on performance. *M* denotes the mean performance of the distributions. Shaded areas show where "improvising" organizations are more successful than "best practice" organizations.

An improvement strategy that simultaneously increases average performance and its reliability is, therefore, not a guarantee of competitive advantage (see *Figure 9.2a*). The consequences of such a strategy is that it helps in competition to avoid relatively low positions, whereas it has a negative effect in competition, where finishing near the top is important. Thus, the price of reliability is a smaller chance of primacy.

If our main goal is to increase the competitive advantage of (small) software organizations, it is time to move away from the model-based, one-size-fits-all thinking of the 1990s. Instead, we should proceed to improvement strategies that focus on learning from our success (Nolan, 1999) to increase the performance distribution's right-hand tail while at the same time reducing the left-hand tail by learning from our failures (Abdel-Hamid and Madnick, 1990; Sitkin, 1992) (see *Figure 9.2b*). As we argued in *Chapter 4* and also in (Dybå, 2000b), improvisation is such an approach to SPI that synthesizes exploitation and exploration and is, thus, of critical importance for success.

Based on existing knowledge and experience, such improvisation builds on Aristotle's concept of *phronesis*: "the ability to spot the action called for in any situation" (Toulmin, 1996, p. 207), and Lévi-Strauss' (1966) concept of *bricolage*: the ability to "make do with 'whatever is to hand'" (p. 17). Seen in this light, *creating an enabling context for SPI can be seen as a socially constructed principle of "regulated" improvisation in which tradition and creativity intersects to generate and interpret new knowledge*.

9.5 Implications for Methodology

In this section, we present important methodological implications of our research, which SPI researchers and consultants could act upon in order to help software organizations improve. Four methodological implications are identified:

- The positivist – interpretivist nexus
- The rigor – relevance nexus
- The convergence – divergence nexus
- The process – practice nexus

The first of these, the *positivist-interpretivist nexus*, was discussed at length in *Section 6.1* and will not be repeated here. The conclusion from that section was that despite seemingly incompatible approaches, it is not only possible but also necessary to apply mixed methods research in ESE. Furthermore, we concluded that mixing qualitative and quantitative research methods should be based upon a *pragmatic* approach. This synthesizes the dialectic between positivism and interpretivism, and offers a practical basis for research in an applied field such as SPI. The *raison d'être* for a pragmatic approach to ESE research is to link theory and practice through mixed method designs and triangulation.

9.5.1 The rigor – relevance nexus

In general, computer scientists tend to adhere to an "ad-hoc" evaluation of their research, publishing ideas with little or no scientific assessment (Fenton *et al.*, 1994; Glass, 1994).

Furthermore, poor theory development and inadequate measurement of constructs seems to be the norm for most studies in SPI (Dybå, 2000a). Additionally, most of the research done so far is failing to influence industrial practice (Potts, 1993). The result of this negligence of both rigor and relevance is what Glass (1994) termed the “software research crisis”.

As we saw in *Chapter 3*, ESE research seeks to address this crisis by encouraging a stronger emphasis on methodological rigor. This is important since rigor is a necessary basis in any research that purports to be relevant. Without rigor, all we have is just “analytical advocacy research”. However, treating ESE as a *natural process* rather than a *social process*, as is too often done, there is a danger that we are looking at it in the wrong way (Pfleeger, 1999a). It is, therefore, a pressing need to avoid that ESE research becomes an Ivory Tower activity with an emphasis on rigor over relevance to practice. As an example of such a growing awareness, the first edition of Fenton’s (1991) book *Software Metrics* had the subtitle *A Rigorous Approach*. In the second edition (Fenton and Pfleeger, 1996), the rigorous framework was still stressed, but the emphasis was now more on its practical application, which was reflected by the change in the subtitle of the book to *A Rigorous and Practical Approach*.

So, *while rigor is a necessary condition for relevant ESE research, it is not sufficient*. Relevant research must also be grounded in industrial projects in order to develop a pragmatic “industry-as-laboratory” approach that narrows the gulf between research and practice (Potts, 1993). Among other things, this means, as we have seen throughout this thesis, a greater emphasis on organizational context. Failure to include such organizational issues may explain some of the inadequacy and dissatisfaction with the existing, model-based SPI approaches; they do not address real organizations. Thus, *to make SPI research relevant, researchers should collaborate with practitioners in real situations and real organizations, identifying relevant problems through close involvement with industrial projects*.

The case studies in TELMET, SPIQ, and PROFIT were organized according to the ESSI PIE model (see Dybå, 2000d), which by now is a well-proven industry-as-laboratory approach to SPI. Furthermore, it is important to note that in this model the researcher is not a neutral observer, detached from practice, as would have been the case in a research approach based purely on positivism. As we saw in *Section 6.1*, positivist research emphasizes the rigorous study of objects without influencing them. In industrial SPI research programs, as in the present investigation, however, an important part of the industry’s motivation to participate, is that the researchers actively participate with relevant knowledge and experience to help the organizations’ improve themselves.

An important implication of this investigation, therefore, is the realization that *action research is particularly well suited as an approach to synthesize rigor and relevance*, and to advance the methodological basis for SPI. There are several reasons for this, as we saw in *Chapter 3*, but the most important is that action research is unique in the way it associates research and practice, so research informs practice and practice informs research synergistically.

9.5.2 The convergence – divergence nexus

As we saw in *Chapter 2*, the rationale behind best practice approaches to SPI is the “convergence hypothesis” (Mintzberg, 1989) of the “one best way”. Such convergent thinking is also found in the engineering traditions of deductive reasoning, which is primarily occupied with finding *the* solution to a convergent problem. However, individuals, teams, and organizational cultures are all different. And, as we have already seen (see *Section 8.4.7*), small successful organizations have capitalized on this diversity and turned it to their advantage. Therefore, when properly understood and imaginatively addressed, diversity offers opportunities for developing better software practices and improving the competitive edge. Ignoring it, however, can result in conflict, disruption, and compromised performance. Thus, our findings suggest that *to be successful in SPI, we need to look beyond the convergence hypothesis, to break away from old routines and standards rather than perfecting existing ones*.

Contrary to convergent problems, divergent problems have no "correct" solution – no matter how much we study them (Schumacher, 1977). It is important to realize this, since divergent problems are not convergent problems that have not yet been solved. Rather, they are problems for which there is no single, best solution. In our experience, most, if not all, problems in SPI are divergent, as for example: "How do we best improve our development process?", "What new products should we develop?", "How can we best satisfy our customers?". To deal with these problems, software organizations and SPI researchers need to acknowledge that a "one-size-fits-all" approach doesn't work in software development and that we should pay more attentions to local needs and specific situations.

The danger of convergent thinking is that it easily may turn SPI methods and "best practices" into a Procrustean bed. In Greek myth, Procrustes offered hospitality to passing strangers, who were invited in for a pleasant meal and a night's rest in his bed. Procrustes described his bed as having the unique property that its length exactly matched whomsoever lay down upon it. What he didn't volunteer was the method by which this "one-size-fits-all" was achieved: a short guest was racked and stretched by Procrustes until he fit; a tall guest, whose head or limbs dangled off the end of the bed, had the overlap chopped off.

This doesn't mean that we should abandon convergent thinking – both convergent and divergent thinking is needed. While divergent thinking is essential for generating a wide range of ideas and seeing new patterns in situations, convergent thinking is necessary to select and evaluate the most useful of the possible solutions. When divergent thinkers prefer to spend more time formulating the problem and exploring creative ideas, convergent thinkers are more concerned about getting a solution under way. Thus, *to be able to take purposeful actions, software organizations, and SPI researchers alike should find the right balance between divergent and convergent thinking in their improvement methods*.

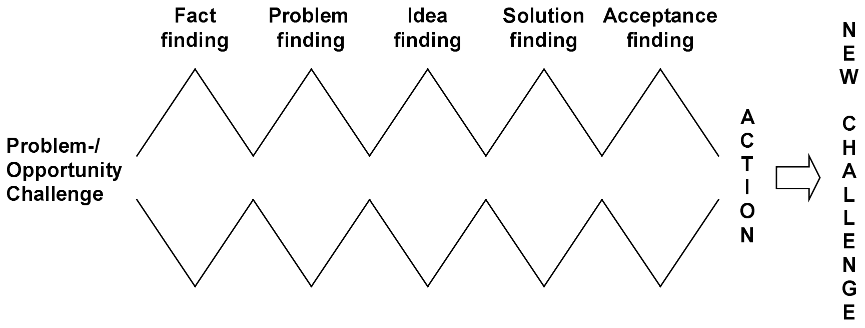


Figure 9.3: The Osborn-Parnes five-stage creative process model (Parnes, 1967).

This is a challenge not only for software organizations, but for academia as well, since, in our experience, the software engineering education emphasizes analytical judgement and convergence at the expense of divergent thinking. Therefore, if creativity in software development and SPI is to be fostered, more emphasis on the divergent process and the interdependence of divergent and convergent thinking is needed. This was also the message behind the July-August 2000 special issue of *IEEE Software* on diversity and in Howard's (2001) analysis of software engineering project management. Furthermore, in systems theory the principle of requisite variety (Ashby, 1956) suggests that the internal control mechanisms of a social system must be as varied as the environment in which it is trying to survive. This way, it can deal with the complexity and challenges of its environment.

In *Chapter 5*, we reported on the effectiveness of creativity techniques, such as the *KJ Method* and *Mind Maps* in generating new organizational knowledge. These techniques, as most problem-solving techniques, are based on the underlying philosophy of Osborn's (1963) "brainstorming" technique. Common to these techniques, is their focus on the dialectic interplay of divergent and convergent activities in the problem-solving process. A problem-solving model that incorporates both divergent and convergent thinking should, therefore, be of methodological interest for anyone purporting to help software organizations taking purposeful actions toward improvements in their development processes (*Figure 9.3*).

9.5.3 The process – practice nexus

A general observation from this study has been that many software organizations have documented their "best practices" as formal procedures, process models, guidelines, rules, check-lists etc. in order to manage and improve their software processes (see e.g. *Section 8.3*). This is in accordance with the prevailing view that formally defining the software process "creates the disciplined and structured environment required for controlling and improving the process." (Florac and Carleton, 1999, p. 6).

However, if we step back, looking at the rules underlying a software group's actions we find that there is an important asymmetry between the rules as represented by process models and formal procedures and the rules as guides in practice. The latter can be viewed in terms of the law of requisite variety (Ashby, 1956): a practice is always richer than any formal representation of it. Also, time-related aspects add to this richness, just like the experience of driving through an area cannot be captured by a map. This richness of experience associated with the social nature of software development is why we at any point in time cannot offer a comprehensive description of its practice. What software development is, at any point in time, depends on how the software developers make sense of it and interprets it to be. Thus, software development is inherently dynamic, indeterminate, and chaotic (Hyman, 1993; Kelsey, 1999; Kraut and Streeter, 1995; Raymond, 1999).

Such indeterminacy of social practice has also been richly illustrated by Orr's (1996) ethnographic study of photocopy repair technicians at Xerox. He studied what reps actually did, not what they were assumed to do. And, as in the case of the reps, what software developers actually do turns out to be quite different from the defined process. This suggests that one size does not fit all when it comes to software teams and their unique needs.

Our results support Brown and Duguid's (1991) perspective on learning-in-working, which suggests that software processes are likely to be ineffective if based on "canonical" rather than "non-canonical" practice. It is not surprising, therefore, that "socialization" and "discussion groups" were among the highest ranked alternatives to formal routines (see *Section 8.3.4*). This is also in agreement with Brown and Duguid's (1991) finding that "story-telling" is of utmost importance for dealing with the complexities of day-to-day practice, much like Zuboff (1988) described story telling as a practice for dealing with "smart" machines.

Thus, contrary to the rigidities of formal processes, stories and the tacit social practices were seen as more flexible, adaptable, and relevant by the software developers in our study. Accordingly, there is an inherent dilemma of balancing the tension between formal process and the informal, social practice for anyone engaged in SPI and in developing SPI methods. This tension between the formal and informal in software development has also been noted by (e.g. Fitzgerald, 1996; Glass, 1995; Wastell, 1996).

While formal process descriptions are useful for simple, routine work, they often fail in the face of turbulence and uncertainty, which typifies most software development. In such situations we cannot use a formal process, simply because we don't know what to do. Indeed, as we argued in *Chapter 4*, we learn what to do and the ways of dealing with it during the actual course of the project. This suggests that improvisation is central in bridging the gap between process and practice, and that those who do the work should be central in designing the process.

Thus, to be successful, our findings (e.g. *Case C*) suggest that *formal processes must be supplemented with informal, inter-personal coordination about practice*. This is consistent with Kraut and Streeter's (1995) conclusions on the subject of coordination in software development. "Process and practice, then, do not represent rival views of the organization. Rather, they reflect the creative tension at the center of innovative organizations." (Brown and Duguid, 2000a, p. 80). Also, and in accordance with Adler and Borys's (1996) discussion of "enabling" and "coercive" bureaucracies, our findings suggest the importance of software processes of the enabling type (see Conradi and Dybå, 2001; Dybå, 2000b).

Furthermore, the inherent lack of conformance between formal process and informal practice is a necessary feature, so that the software organization may learn from its theories of action. Deeper, second-order or double-loop learning not only requires that the software organization takes actions to correct errors, but also that it critically reexamines the validity of its basic norms and assumptions and, as a result, is prepared to modify existing strategies. Thus, in the extreme case of complete process conformance, learning and, hence, SPI would not be possible (see *Figure 4.8*).

Rather than detailed process descriptions, this suggests that software processes should be described according to the "rule of bifurcation" (Armour, 2001), supporting different degrees of formality. I.e., they could be stated in terms of two levels: a general description of the process, supplemented by specific examples from practice. Like the tale of the sitar player asking Buddha how best to tune his instrument, therefore, we find the famous answer: "*not too tight, and not too loose*" as a good general rule for balancing formal process with informal practice.

9.6 Implications for Practice

In this section we present a set of enabling conditions, or guidelines, that practitioners can adopt to implement a successful SPI program in their organizations. We do not promise that this involves a straightforward and simplistic process. However, *the guidelines offered have grown out of our empirical investigations, thus, guaranteeing that they are grounded in the "real" world*. First, we offer some guidelines for building a learning software organization. Next, we suggest a performance management model for SPI. Finally, we offer some specific advice for managers and developers in software organizations that want their organizations to excel.

9.6.1 Building a learning software organization

As our investigations have shown, learning is not only central to the activities of software development, it is the *sine qua non* of software process improvement. Thus, a major implication for practice is to utilize the insights gained from these investigations to build a learning software organization. Based on the results from this study, therefore, we suggest the following seven essential guidelines for building a learning software organization:

1. ***Share knowledge and experience.*** Knowledge and experience are among the primary preconditions for successful SPI. Software companies that want to succeed with SPI must learn to value, trust, and rely on the collective insights of their people to create a learning software organization. More importantly, the organization should enable SPI by creating a context in which the sharing of knowledge and experience is valued. This presupposes a system of organization-wide sharing – an *organizational memory*. Furthermore, it presupposes a system of incentives so that the members of the organization can build social capital by sharing their knowledge and experience.

   Up until now software organizations have been advised to populate their experience bases with large amounts of explicitly available knowledge such as quantifiable facts, formulas, routines, and procedures. While this is still important, the future learning software organizations must create environments where experiential knowledge can be shared through dialogue and interaction. There is, thus, a need to acknowledge how tacit knowledge supports quantifiable data, thereby providing the software organization with a more complete, *situated learning* system.

2. ***Support innovation and creativity.*** Along with knowledge and experience, successful SPI depends on innovation and creativity. These characteristics require an organizational culture that continually questions and tests the organizations shared beliefs and assumptions. They call for an environment that actively supports experimentation, risk-taking, and failure.

   Like Csikszentmihalyi (1996) found that *luck* was the most often-mentioned reason for successful creativity, so too are software organizations beginning to acknowledge that many of their successes were the result of *probing and learning*, rather than brilliant foresight. The derivation of 3M's Post-it Notes is a classic tale of such accidental success and a good illustration of how the innovative capacity of a firm depends on its members' efforts to alleviate tensions between exploitation of existing knowledge with exploration of new knowledge.

3. ***Situate learning in real work.*** Many organizations view learning as a stand-alone activity separate from daily work. Employees are trained away from their workplace, and then are assumed to perform their jobs according to the procedures learned during training. However, *software development is the core process for knowledge and experience sharing, innovation, and creativity in a software organization*. How well the organization manages its development process is, therefore, a critical factor in determining how successful its organizational learning and SPI activities can be carried out.

   The organization's core competence is more than the explicit "know-what" that can be shared by several. A software organization's core competence requires the more elusive "know-how" (Ryle, 1954) – the particular ability to put know-what into practice. It is this know-how, which is embedded in the organization's collective work practices that is key to organizational learning and SPI. Thus, *developers learn as they work and the software organization learns from the collective experience of its developers*.

4. ***Launch self-organizing teams.*** Knowledge and experience sharing, innovation, and creativity thrive best in small groups where people can interact freely. A few individuals with new ideas and creative thoughts are not enough for building a learning software organization. Thus, the software organization must encourage *communication*, *coordination*, and *collaboration* within and between software teams, ensuring that they are flexible enough to form, change, and dissolve, as needed.

   Software teams should be provided with enough autonomy that they can act on what they observe and learn, with enough coordination that their local actions work together toward the organization's core values rather than producing sub-optimizations. The coordination between such teams is achieved through the organization's memory by collectively generating and interpreting knowledge.

   This autonomy can best be achieved through a decentralized structure that makes teams or small business units self-organized and self-managed. Also, team size should not be larger than what is practical for coordination by *mutual adjustment*, since this is the only coordination mechanism that can handle really complex work with a lot of problem solving (Mintzberg, 1989). Through such autonomy, the organization may increase the chance of introducing "unexpected opportunities" (Nonaka and Takeuchi, 1995).

5. ***Manage diversity.*** A key ingredient in successful software teams is diversity. Homogeneous groups and uniform processes promoted by "best practice" models tend to produce single-loop learning. To foster innovation and creativity, it is necessary to bring together diverse groups of people with different levels of expertise and a broad spectrum of ideas to tailor processes to meet specific local needs.

   Diversity of perspectives is essential to the creation of new collective knowledge as well as to developing new perspectives on existing knowledge. Furthermore, since individuals, teams, and organizational cultures are all different, a "one-size-fits-all" approach doesn't work in software development and SPI. Therefore, when properly understood and imaginatively addressed, diversity offers opportunities for developing better software practices and improving the competitive edge.

6. ***Establish strong core values.*** While autonomy and diversity are important enablers for the learning software organization, there must be some grounding entity that unites the diverse and independent teams and their efforts. Traditionally, this entity has been

managerial control. In the learning software organization, that entity is “business orientation”, or core values. These values allow for coordination without control and for experimentation and adaptation without anarchy. The organization’s core values create a sense of purpose – they enable the software organization to focus its attention and provide a direction despite a seemingly random behavior.

Furthermore, *core values provide the most important criteria for the justification of new organizational knowledge*. In the absence of such core values, it would be impossible to decide whether new concepts and beliefs are worthy of further attention and investment. Therefore, the commitment of organizational members to organizational learning and SPI depends critically on explicitly formulated strategies and values.

7. ***Collectively reflect in and on practice.*** Critical reflection is important in any form of learning (Kolb, 1984; Schön, 1983). Also, the benefits of the “creative chaos” induced by diversity and local autonomy can only be realized through reflection upon organizational actions; without such reflection, ambiguity and fluctuation tend to lead to “destructive” chaos (Nonaka and Takeuchi, 1995).

   Similar to the ancient Roman forum, which was a central gathering place where citizens discussed the great issues of the day, the learning software organization is equally in need of settings where it can collectively reflect *on* its practice. Such *learning forums* can take many forms. They include process assessments, survey feedback, internal benchmarking projects, and postmortem reviews, just to mention a few. Common to these learning forums is that a small group of people is assembled with a well-defined learning agenda and a compressed learning cycle in order to improve their collective understanding of the issues of interest. Also, when organizational members encounter a problem, there is a need for collective reflection *in* practice, for example by calling a meeting and getting immediate feedback. This is important, since without action and short feedback cycles, organizational members’ interest in SPI will rapidly disappear.

9.6.2 Performance management

In *Chapter 5* we argued that organizational performance is the ultimate criterion for any SPI activity. That is, we must ensure that gains from SPI activities have in fact been made with respect to organizational behavior and not merely at the cognitive level. Consequently, we focused on organizational performance as the dependent variable in this. However, measuring organizational performance is not only important for the present investigation, performance management should be an integral part of any industrial organizational learning and SPI initiative.

Therefore, based on the findings from this study, and inspired by Cummings and Worley (2001), we suggest a performance management model for SPI, which includes activities for goal setting, data collection, data analysis, and feedback (see *Figure 9.4*). Also, based on the finding that feedback is the most important part of a measurement-based SPI approach (e.g. *Case A* and Dybå 2000e), these activities jointly influence the performance of software teams and organizations. As we have already seen, timely and accurate feedback is essential for guiding successful SPI actions. Thus, the real payoff from using data-based methods in the organizational learning cycle comes when the data are fed back to the local teams, from which it was collected, and problem solving begins.

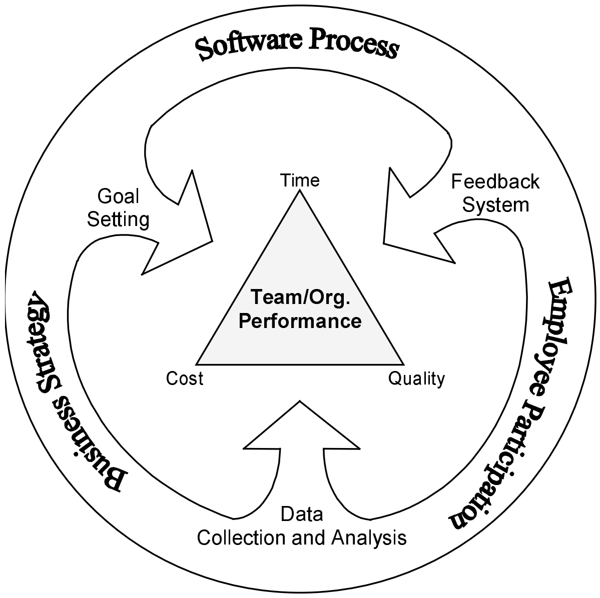


Figure 9.4: SPI performance management model.

The usefulness of the organization's feedback system in guiding purposeful actions is largely determined by the processes used to collect the data, the techniques used to analyze the data, and the way in which the feedback data are interpreted. If the wrong data is collected, if data is inadequately analyzed, or if the process of data collection creates suspicion and anxiety, then successful feedback will be very difficult to obtain. Conversely, data collection methods and practices that generate interest and thought about important problems, collection techniques that enable gathering of useful and important data, and analysis which enables meaningful interpretations of those data, can greatly aid feedback (Nadler, 1977).

The organization's software processes are both the source and the target for SPI. It is during the execution of these processes that knowledge is created. And, as with organizational learning and SPI in general, the way in which these processes are handled is a critical factor for effective performance management. It is, therefore, particularly important that the software organization considers its internal software development processes as the foundation for learning. However, since exploitation generally generates clearer, earlier, and closer feedback than exploration, it is important that the organization balances learning from experience with innovation and creativity so that exploitation does not drive out exploration during the measurement process.

The software organization's business strategy should guide the definition of the goals and objectives that are needed to decide what kind of data to collect, how to collect it, how to use it, and how to evaluate it. As a practical aid in such goal setting, we recommend using the templates developed specifically for this purpose in GQM (see *Chapter 2*). Together with business orientation, employee participation is, as we have seen, the most critical factor for successful SPI. Their joint contribution explains about half of the variance in SPI success (adjusted R-square = 49.2, F = 58.57, p < 0.0005), see *Table 8.17* and *Table 8.18*. This suggests that employee participation also is an essential factor in determining the nature of the organization's performance management practices. Software developers and managers should, therefore, participate in all the main activities of the performance management model: goal setting, data collection and analysis, and feedback.

To summarize, performance management is an integral part of the learning software organization. Its focus should, therefore, be to enable the software organization's processes of sensemaking, knowledge creation, and purposeful actions. Our findings thus suggest that the software organization's performance management model should have the following properties:

- A focus on a few metrics that are critical to organizational success and in supporting the organization's business strategy and culture.
- A balance between outcome measures that assess performance and process measures that enable learning.
- A steady stream of measurements from an ongoing process of learning from experience and experimentation.
- A collective feedback system that closes the learning cycle and helps turn knowledge about SPI into organizational action.

9.6.3 Leading organizational learning and SPI

The fundamental role of managers and leaders is to shape and create organizational contexts that are supportive of and conducive to organizational learning and SPI. To the extent that the software organization remains locked into the old context, no significant change, or improvement is possible. This means that managers have to become skilled in enabling the learning processes and key factors for success, in order to define new and appropriate contexts for SPI. In this way, they can help to shape the emergent processes of self-organization in software teams by providing "minimum specifications", while avoiding the trap of imposing too much control.

The manager can create such contexts by enabling a belief-driven process that generates *new understandings* of a situation, or by engaging in *new actions* through an action-driven process. The conventional way of thinking about SPI puts these in a sequential order, i.e. first understanding, then action. For example, in the early Japanese software factories, a strong emphasis was put on gathering data on existing processes before changing or improving them (Cusumano, 1991). Likewise, Basili *et al.* (1994b) and Basili and Caldiera (1995) argued that we should first understand what we do, before we attempt to change it. From a learning perspective, however, actions and understandings often need to be reversed. Therefore, the generation of new understandings and new actions, in whatever order they evolve, are fundamental in creating the contexts that enable SPI.

Congruent with the results of our investigations, we suggest the following five guidelines for leading organizational learning and SPI:

1. ***Create a knowledge vision.*** Top management should create a knowledge vision and communicate it within the organization. The vision should define the context and provide a general direction regarding what kind of knowledge organizational members ought to seek and create. This is important in order to ensure a well-founded business orientation

in organizational learning and SPI. The essence is, thus, to enable the organization to take collective action, generate new understandings, and make them available to the rest of the organization.

2. ***Create learning opportunities.*** The time pressure and workload in most software organizations makes the urgent drive out the important. Organizational learning and SPI is easily postponed in the face of more immediate demands. For learning to occur, the organization's managers and leaders must, nevertheless, ensure that enough time and space are set aside for reflection and the sharing of knowledge. Also, a critical task of the manager is to assure that the organization's feedback process is open, healthy, and effective, so that efforts can be directed toward problem solving, new opportunities, and constructive change, rather than denial, defensive behavior, or misuse of data.

3. ***Involve everyone.*** Together with business orientation, employee participation is the most critical factor for success in SPI. Therefore, software organizations should capitalize on the diverse talents of their members and turn it to their advantage. This means that everyone should be involved in the organization's learning process. Thus, ask your employees how you best can help them – listen and respond. Ignoring this might easily result in conflict, disruption, and compromised performance. Remember: people tend to support what they have participated in creating.

4. ***Start with the obvious.*** Don't search outside the organization for a silver bullet. SPI is not about seeking expert answers from outside the organization. As Dorothy discovered in *The Wizard of Oz* (Baum, 1995), there's no place like home. Rather than searching for the latest fad, start with the obvious, "hot spots", within your own organization. Remember that the most powerful learning comes from firsthand experience. Therefore, use the organization's software projects as the learning lab and situate organizational learning and SPI in these projects.

5. ***Tolerate failure.*** If experimentation, risk-taking, and trial-and-error modes of problem solving are to be valued, then the organization's culture must tolerate failure and refrain from placing blame. Rather than treat failure as unacceptable and stigmatizing, the organization should distinguish between failures that result from carelessness and failures that are a result of intelligent efforts to experiment outside existing patterns (Barrett, 1998). In this way, the manager can draw attention to potential problems and stimulate the search for creative and innovative solutions when the environmental factors are changing. *The paradox is that you have to experience failure in order to have success.*

   The following story about Thomas Watson Sr., IBM's founder and CEO for many decades illustrates this (Bennis and Nanus, 1997):

   > A promising junior executive of IBM was involved in a risky venture for the company and managed to lose over $10 million in the gamble. It was a disaster. When Watson called the nervous executive into his office, the young man blurted out, "I guess you want my resignation?" Watson said, "You can't be serious. We just spent 10 million dollars educating you" (*ibid.*, p. 70).

9.6.4 The responsibilities of organizational members

Organizational knowledge is created by the ongoing experience and experimentation of organizational members. Therefore, all members of the organization should consider their responsibilities and roles in building a learning software organization. Without individuals who learn, there can be no organizational learning. Software developers, for example, generate, accumulate, and interpret both tacit and explicit knowledge as part of their actions to develop quality software. Many of them work in direct contact with the customer, which means that they are the software organization's front line of business. Thus, being at the "gemba" (Zultner, 1993), they are in a position to obtain the latest information on the developments in the market, technology, or competition.

Thus, we offer the following five responsibilities that members of a (future) learning software organization should consider:

1. ***Reflect in and on practice.*** Engaging in a process of action and reflection is an important part of individual learning. Therefore, organizational members should pursue a "reflective" attitude about their own, individual work practices. Reflection is a way of bringing concrete experience into abstract conceptualization. It is a way of investigating the possible interpretations of what is or what was experienced. Therefore, it is important to learn more about ones professional practice by reflecting *in* the ongoing practice as well as *on* the practice after some action has been taken. Reflection on practice can be considered as ordinary experiential learning, while reflection in practice is a knowing process, i.e. the simultaneous process of acting and reflecting. Thus, by reflection, organizational members can take responsibility for trying new ways of doing things and to learn in and from their actions.

2. ***Actively engage in dialogue.*** Successful SPI is closely connected to both organizational *means* and organizational *ends*. Actively engaging in a public dialogue about organizational means and ends, and their relationships, is important for any learning initiative to succeed. Organizational members should, therefore, take responsibility to make individual reflections accessible to other organizational members, and openly question or challenge the organization's underlying strategies and assumptions. A way of doing this is by taking the initiative to create learning forums.

3. ***Be open to new perspectives.*** History has taught us that even the most established truths must eventually be revised or replaced. For individuals, groups, and organizations to learn, this requires an openness that accepts the provisional nature of knowledge. Good indicators of such openness are one's attitude toward challenging questions, one's willingness to refine one's theories-in-use and assumptions about the way the world works, and one's willingness to seek feedback from others. Remember: Breakthroughs occur when theories are overturned, updated, or replaced.

4. ***Take responsibility for sharing knowledge.*** All members of the organization must take responsibility to make what they have learned available to others. The traditional practice of hoarding knowledge in order to enhance one's personal power is not only unacceptable – it will backfire. In the learning software organization, power accrues to those who become a source of knowledge by sharing what they know.

5. ***Help colleagues succeed.*** Since software development is a team effort, organizational members succeed only as their colleagues succeed. In many ways, the role of the developer resembles that of a soccer player. While each player has a fixed position, the whole team moves together while individual members retain their relative positions. Just like a soccer team moves with the ball, the software team moves with the task. This suggests that rather than internal competition, organizational members should take responsibility to foster an atmosphere of collaborative and cooperative work efforts.

9.7 Chapter Summary

In this chapter, we have discussed the answers to the three research questions posed in *Chapter 1*. First, with respect to the learning processes in SPI we discussed the implications of the finding that software developers *actively* construct knowledge in their attempts to make sense of their world. This led to a focus on the dialectic between local communities of practice and living organizational memories. Next, we discussed the findings regarding the importance of each of the six key factors of success in SPI identified in our model. Then we discussed the effects of organizational context on organizational learning in software organizations; particularly the differences found between small and large software organizations in unstable and changing situations.

Furthermore, the implications for theory and methodology were discussed using a dialectic approach. With respect to theory, four relationships were discussed: the tacit-explicit nexus, the local-global nexus, the knowledge-action nexus, and the exploration-exploitation nexus. With respect to methodology, we discussed the positivist-interpretivist nexus, the rigor-relevance nexus, the convergence-divergence nexus, and the process-practice nexus. Finally, the implications for practice were discussed in terms of practical recommendations for successful SPI in terms of guidelines for building a learning software organization, for performance management, and for the roles and responsibilities of managers and developers.

In the next and final chapter, we present our conclusions on the research problem and questions, and state the claimed contributions of the thesis. Furthermore, we make an evaluation of the thesis with respect to both rigor and relevance. Finally, we describe the limitations to the study along with recommendations for future research.

Chapter 10

Conclusions

"Alice: 'Would you tell me, please, which way I ought to go from here?'
'That depends a good deal on where you want to get to,' said the Cat."

– Lewis Carroll

In the previous chapter we discussed the answers to the three research questions posed in *Chapter 1*. First, with respect to the learning processes in SPI we discussed the implications of the finding that software developers *actively* construct knowledge in their attempts to make sense of their world. Next, we discussed the findings regarding the importance of each of the six key factors of success in SPI identified in our model. Then we discussed the effects of organizational context on organizational learning in software organizations. Furthermore, the implications for theory and methodology were discussed using a dialectic approach, while the implications for practice were discussed in terms of practical recommendations for successful SPI.

In this final chapter, we present our conclusions on the research problem and questions, and state the claimed contributions of the thesis. Furthermore, we make an evaluation of the thesis with respect to both rigor and relevance. Finally, we describe the limitations to this study along with recommendations for further research.

10.1 Conclusions on the Research Problem and Questions

The fundamental assumption behind the investigation presented in this thesis was that *software process improvement is a socially constructed learning process* and, consequently, that a commitment to learning rather than to "best practice" models is needed to accomplish improvements in software development.

The position taken regarding organizational issues was strongly influenced by socio-technical theory and its central conception that organizations are both social and technical systems, and that the core of the software organization is represented through the interface between the technical and human (social) system. Accordingly, we defined organizational issues within the context of SPI as *any distinct area on the interface between the technical system and the social system, which can enable software process improvements*.

Moreover, despite the increasingly recognized importance of organizational issues within the SPI community, few empirical studies have been reported. To help fill this gap, the doctoral study, reported in this thesis, was initiated to explore the relative importance of organizational issues by investigating the learning processes and key factors for success in SPI. Thus, the research problem addressed in this thesis was:

How can software organizations enable software process improvement?

To narrow the focus of the investigation, the research problem was addressed by studying three research questions. So, in order to provide an answer to the research problem, we first present the conclusions to each of these questions.

Q1: What are the key learning processes in successful software organizations?

The first question focused on identifying the key processes that are part of a successful software organization's learning cycle. In addition to our own practical experience of software development, a theoretical investigation and a set of qualitative case studies were conducted to answer the question.

In addition to identifying SPI as a social, collaborative activity, these investigations also led us to the identification of sensemaking, knowledge creation, and purposeful actions as the foundations of the learning software organization. Therefore, to explain why some SPI initiatives are more successful than others are, we constructed a dynamic model of SPI that incorporated these foundations.

The key to our dynamic model of SPI was an understanding of the dialectic nature of the learning processes and their enabling conditions. Most importantly was the recognition of the dialectical process between generation of new knowledge, through action, resulting in organizational knowledge; and the interpretation of this organizational knowledge, through sensemaking, into local knowing.

The main conclusion on the first research question is that *the key to successful learning is a continuous and simultaneous dialectic interplay between the knowledge that the organization has established over time, and the knowing of the organization's members in their respective contexts*. Thus, recognizing the fundamental principle of the hermeneutic circle, successful learning requires the joint creation of local and organizational knowledge.

Q2: What are the key factors of success in software process improvement?

The second question focused on identifying the key factors for success in SPI. In addressing this question we performed a theoretical study and a quantitative survey to investigate the key factors of success in SPI.

The results of the bivariate correlational and multiple regression analyses indicated support for all of the hypotheses in the proposed theory and demonstrated the existence of important factors for SPI success. These findings add an important new dimension to ESE research in that they verify the importance of organizational factors for SPI success. This suggests that, rather than trying to imitate technical procedures, software organizations should focus their SPI efforts on creating an organizational culture within which these procedures can thrive. This differs substantially from that found in most of the existing SPI literature, which focuses almost entirely on software engineering tools and techniques.

The main conclusion on the second research question is that *SPI success depends critically on six organizational factors: business orientation, involved leadership, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge*. Together, these factors explained 56 percent of the variance in SPI success (adjusted R-square = 0.56, $F = 25.95$, $p < 0.0005$). Furthermore, the results of the multiple regression analysis showed that employee participation ($\beta = 0.297$, $t = 3.80$, $p < 0.0005$) and business orientation ($\beta = 0.245$, $t = 2.78$, $p < 0.01$) were the most important factors in predicting SPI success. In fact, these two factors explained 49 percent of the variance in SPI success (adjusted R-square = 0.49, $F = 58.57$, $p < 0.0005$).

Q3: What are the relationships between organizational context and modes of learning in software organizations?

The third question focused on finding the relationships between organizational context and modes of learning in software organizations. Two modes of learning were identified in this study: exploitation of existing knowledge and exploration of new knowledge. Furthermore, we identified two contextual variables to capture the most influential sources of variation: environmental turbulence and organizational size. Thus, to answer the third research question, we examined the effects of organizational size and environmental conditions on the learning modes by comparing the estimated marginal means of exploitation of existing knowledge and exploration of new knowledge, using two-way ANOVA with organizational size and environment as the two classification variables.

Two important results emerged from the investigation of this research question. First, the results showed that environmental turbulence had no influence on the level of SPI success. What's more important, however, is that large successful and small successful organizations differed fundamentally in their respective approach to SPI. The results showed that small software organizations engaged in significantly ($p < 0.001$) more exploration in turbulent environments than large software organization, which suggests that *the main difference* *between small and large software organizations is the ways in which they react to unstable and changing stimulus situations*.

The most important conclusion on the third research question is that *small software organizations in turbulent environments, require learning strategies that are closely aligned with explorative behavior, while at the same time promoting the exploitation of past experiences*. Rather than "best practice" approaches such as the CMM, which rely on a predictable sequence of events, this suggests an SPI strategy based on improvisation in which organizational members prepare their organizations to learn while in the process of acting.

Taken together, the answers to these three research questions constitute the main contributions of this thesis. So, returning to the original question on how software organizations can enable SPI, we found that technology is not enough to solve the difficult problems facing most software organizations. Knowing is a human act. Therefore, in addition to new technologies, software organizations must also support social interaction among and within local teams in order to create enabling contexts for SPI. In other words, *software organizations that want to prosper in the 21st century should synergistically combine technology with social collaboration to become learning software organizations*.

To create such learning software organizations, we have proposed a number of dialectic relationships. A fundamental understanding of these relationships are necessary to create the contexts needed to enable SPI. However, instead of thinking of e.g. knowledge vs. action, exploration vs. exploitation, rigor vs. relevance, or process vs. practice, it would be more useful to think of how the two sides of the dialectic are interlaced. From this perspective, it will be more important to think in terms of communities of practice, improvisation, living organizational memories, and action research to enable SPI. Then again, some of the dialectic relationships are obviously more difficult to synthesize than others. In some cases, a dynamic synthesis may be impossible to obtain. Nevertheless, it is precisely in the dialogues and interactions of searching for such syntheses that enabling contexts are created.

Thus, to create a learning software organization, a different context than that proclaimed by current "best practice" SPI approaches must be created. The conclusion on the research problem is that *SPI cannot be managed, but only enabled through the space in which the software organization creates the possibilities for sensemaking, knowledge creation, and purposeful action.*

10.2 Contributions

The objective of the thesis was to investigate the importance of organizational issues in SPI. Based on this objective and the answers provided to the three research questions posed in *Chapter 1*, the thesis contributes with unique theoretical, methodological, and practical knowledge.

For the most part, SPI has been preoccupied with technical issues at the expense of organizational issues. Therefore, *the principal contribution of the thesis is to increase the understanding of the influence of organizational issues in SPI by empirically showing that they are at least as important as technology for succeeding with SPI.*

10.2.1 Theoretical contributions

We have made several theoretical contributions throughout the thesis. The most important are:

- **Foundations of the learning software organization**. One of the key assumptions in the thesis was that successful SPI presupposes an orientation toward organizational learning. Based on this assumption we made a theoretical contribution by defining the foundations of the learning software organization (see *Chapter 4*).
- **A dynamic model of SPI**. Based on personal experience and observations, and the extensive literature review, we contributed to the theory of SPI by developing a dynamic model of SPI. The model is inspired by social constructivism and open systems theory. It consists of the following four major elements: organizational context, learning cycle, facilitating factors, and organizational outcome. The model was presented in *Chapter 5*, and provided the answer to research question *Q1*.
- **Key factors for success in SPI**. As an important part of the dynamic model of SPI, we contributed to the theory of SPI by defining a set of key factors for success. The importance of these factors was validated through the quantitative survey. The factors were defined in *Chapter 5* and validated in *Chapter 8*, and provided the answer to research question *Q2*.
- **An improvisational approach to SPI**. The results of the investigations showed that there are important differences in the way large successful and small successful organizations react to environmental conditions and how they deal with SPI. Consequently, we proposed an approach to SPI in small organizations based on improvisation. This contribution was described in *Chapter 9*, and provided the answer to research question *Q3*.

10.2.2 Methodological contributions

The main methodological contributions of the thesis are:

- **A pragmatic approach to ESE research**. For the most part, ESE research has been concerned with experimental and quantitative methods. We claim that all research is a social construction. Consequently, we defined a pragmatic research approach, integrating positivist and interpretative approaches, which we believe are generally applicable to ESE research. The foundations for this approach were described in *Chapter 6*.
- **An instrument for measuring the key factors of success in SPI**. Instrument validation and reliability issues have been inadequately addressed in ESE research. By applying psychometric theory, we developed an instrument for measuring the key factors of success in SPI. The instrument was shown to have desirable psychometric properties. The process of developing the instrument was described in *Chapter 7*, while the instrument itself is included in *Appendix E*.

10.2.3 Practical contributions

Based on the results of the investigations in this thesis, we have made several contributions for practice. The main contributions are (see *Chapter 9*):

- **Guidelines for building a learning software organization**. Our investigations have shown that learning is the *sine qua non* of SPI. Based on the insights gained, we contributed with a set of essential guidelines for building a learning software organization.
- **A performance management model for SPI**. Throughout the thesis we argued that organizational performance is the ultimate criterion for any SPI activity. Based on our findings, therefore, we made a contribution to practice by proposing a performance management model, which includes activities for goal setting, data collection, data analysis, and feedback.
- **Guidelines for leading SPI**. Congruent with our findings and the foundations of the learning software organization we suggested a set of guidelines for creating the appropriate contexts for organizational learning and SPI in an industrial setting.
- **Responsibilities of organizational members**. For organizational learning to take place, individual members of the organization must take on certain responsibilities. Consequently, we proposed a set of roles and responsibilities in building a learning software organization that organizational members should consider.

10.3 Evaluation

Throughout the thesis we have argued that SPI research should become more rigorous and more relevant. We have argued that methodological rigor is an important and necessary basis in any research that purports to be relevant. In addition, for SPI research to be relevant it must also be grounded in real industrial needs in order to narrow the gulf between research and practice.

Consequently, we take our own medicine and evaluate the main, quantitative part of the present investigation according to both rigor and relevance. Methodological rigor is evaluated according to the guidelines proposed by Kitchenham *et al*. (forthcoming) (see *Table 10.1*). Relevance is evaluated according to Benbasat and Zmud's (1999) recommendations for increasing the relevance of IS research (see *Table 10.2*).

Table 10.1: Evaluation of rigor according to Kitchenham *et al*.'s (forthcoming) guidelines for empirical research in software engineering.

Criterion	**Compliance**
Experimental context:	
1. Be sure to specify as much of the industrial context as possible. In particular, clearly define the entities, attributes and measures that are capturing the contextual information.	Yes, context information was described in *Sections 1.2* and *6.3*. The specific measures used to explore the effects of contextual variables were specified in *Table 6.3*.

Criterion	Compliance
2. If a specific hypothesis is being tested, state it clearly prior to performing the study, and discuss the theory from which it is derived, so that it implications are apparent.	Yes, *Section 5.5* was fully dedicated to the discussion of the theory from which the seven hypotheses in the study were derived.
3. If the research is exploratory, state clearly and, prior to data analysis, what questions the investigation is intended to address, and how it will address them.	Yes, in *Section 1.3* we stated the research problem and questions and how they would be addressed in this study.
4. Describe research that is similar to, or has a bearing on, the current research and how current work relates to it.	Yes, the extensive literature review presented throughout the thesis describes similar work. Also the final discussion relates this study to other work. See *Chapters 2-5*, and *9*.
Experimental design:	
5. Identify the population from which the subjects and objects are drawn.	Yes, the population from which the subjects were drawn was identified in *Section 6.3*.
6. Define the process by which the subjects and objects were selected.	Yes, the sample selection process was described in *Section 6.3*.
7. Define the process by which subjects and objects are assigned to treatments.	Not applicable.
8. Restrict yourself to simple study designs or, at least, to designs that are fully analyzed in the literature.	Yes, we used a sequential, mixed method design that was justified by reference to the research literature. See *Section 6.2*.
9. Define the experimental unit.	Yes, the unit of analysis was defined in *Section 6.2* and also in *Section 1.5*.
10. For formal experiments, perform a pre-experiment or pre-calculation to identify or estimate the minimum required sample size.	Yes, power analysis and calculation of the minimum required sample size were done in *Section 6.7*. NB! This is just as important for a survey as it is for a formal experiment.
11. Use appropriate levels of blinding.	Not applicable.
12. If you cannot avoid evaluating your own work, then make explicit any vested interests (including your sources of support), and report what you have done to minimize bias.	Yes. In the qualitative part of the study, we actively participated in the improvement processes. The influence of this participation for the collected data was described in *Sections 6.4* and *6.5*.
13. Avoid the use of controls unless you are sure the control situation can be unambiguously defined.	Yes, organizational size and environmental conditions were unambiguously defined and included for quasi-experimental control. See *Section 5.2*.
14. Fully define all treatments (interventions).	Not applicable.
15. Justify the choice of outcome measures in terms of their relevance to the objectives of the empirical study.	Yes, the outcome measure SPI success was defined and justified in *Section 5.4* and operationalized in *Section 7.9*. Also see *Appendix E*.

Criterion	Compliance
Conducting the experiment and data collection:	
16. Define all software measurements fully, including the entity, attribute, unit and counting rules.	Yes, all aspects of the instrument developed for data collection was described and justified in *Chapter 7*.
17. For subjective measures, present a measure of inter-rater agreement, such as the kappa statistic or the intra-class correlation coefficient for continuous measures.	Yes, reliability and validity measures were reported in *Sections 7.7-7.9* according to standard psychometric principles and criteria.
18. Describe any quality control method used to ensure completeness and accuracy of data collection.	Yes, completeness and accuracy of the data were checked continuously as questionnaires arrived during the data collection period.
19. For surveys, monitor and report the response rate, and discuss the representativeness of the responses and the impact of non-response.	Yes, the response rate was reported and discussed together with the representativeness of the responses. See *Section 6.3*.
20. For observational studies and experiments, record data about subjects who drop out from the studies.	Not applicable. Given the high response rate in this study, no further analysis was done on the differences between respondents and non-respondents. See *Section 6.3*.
21. For observational studies and experiments, record data about other performance measures that may be adversely affected by the treatment, even if they are not the main focus of the study.	Yes, a balanced set of performance measures and perceived levels of success was included in the study. See *Sections 5.4* and *7.9*. Also see *Appendix E*.
Analysis:	
22. Specify any procedures used to control for multiple testing.	Not applicable.
23. Consider using blind analysis.	Not applicable.
24. Perform sensitivity analyses.	Yes, sensitivity analyses for all variables were performed and reported in *Section 8.4.1*.
25. Ensure that the data do not violate the assumptions of the tests used on them.	Yes, all assumptions of the tests used on the data were evaluated in *Section 8.4.1*.
26. Apply appropriate quality control procedures to verify your results.	Yes, graphical examinations and exploratory data analysis were done prior to detailed analysis.
Presentation of results:	
27. Describe or cite a reference for all statistical procedures used.	Yes, the procedure for hypothesis testing was described in *Section 6.7*. Furthermore all but the most basic tests were explicitly referenced.
28. Report the statistical package used.	Yes, the data analysis techniques and statistical package used was reported in *Section 6.6*.

Criterion	Compliance
29. Present quantitative results as well as significance levels. Quantitative results should show the magnitude of effects and the confidence limits.	Yes, quantitative results and significance levels were reported for all statistical tests. See *Chapter 8*.
30. Present the raw data whenever possible. Otherwise, confirm that they are available for confidential review by the reviewers and independent auditors.	Yes, the raw data of the survey is included in *Appendix G*.
31. Make appropriate use of graphics.	Yes, results from two-way ANOVA tests were presented graphically; for the other tests, simple tables were used to report results.
Interpretation of results:	
32. Define the population to which inferential statistics and predictive models apply.	Yes, the population was defined in *Section 6.3*.
33. Differentiate between statistical significance and practical importance.	Yes, the relationships between statistical significance and practical importance were discussed in *Section 6.7*.
34. Define the type of study.	Yes, the type of study was defined in *Section 6.2*.
35. Specify any limitations of the study.	Yes, the limitations of the study are specified in *Section 10.4*.

Table 10.2: Evaluation of relevance according to Benbasat and Zmud's (1999) recommendations for increasing the relevance of information systems research.

Criterion	Compliance
Topic Selection:	
1. Focus on future interests of key stakeholders	Yes, the importance of organizational issues in SPI is getting increasingly more attention among colleagues and practitioners, and also in practice oriented magazines such as *IEEE Software* and *Communications of the ACM*.
2. Identify topics from software development practice.	Yes, our industrial background and continuous collaboration with practitioners during the last 15 years have revealed that organizational issues are considered important for industrial SPI programs.
3. Identify, as an academic community, the core research issues that can influence practice in the future	Yes, the current investigation is part of an effort to reach some degree of consensus as a community regarding the core phenomena associated with relevant SPI research.

Criterion	**Compliance**
Purpose:	
4. Focus on the likely outcome (that can influence practice) rather than on inputs (academic and intellectual challenges) when choosing a research project.	Yes, in *Section 1.1* we pointed out that the focus of the study was on how to succeed with the practice of SPI. This is also seen in the implications for practice in *Section 10.6*.
5. Develop cumulative, theory-based, context-rich bodies of research to be able to make prescriptions and be proactive	Yes, the foundations of the learning software organization (*Chapter 4*) and the dynamic model of SPI (*Chapter 5*) are such theories. Also, the measurement instrument (*Appendix E*) developed in *Chapter 7* provides reliable and valid measures of the key factors for success in SPI.
6. Develop frames of reference to organize phenomena and provide contingency approaches to managerial action	Yes, the learning cycle and the key factors for success in SPI (*Chapter 5*) are the two most important frames of reference developed in this study. Also, the implications of blindly following "best practice" approaches to SPI, is such a frame of reference. See *Section 9.4.4* and Dybå (2000b).
7. Portray research outputs in ways practitioners can utilize to justify and rationalize SPI related decisions	Yes, in *Section 9.6* we offered guidelines for building a learning software organization. Also, we suggested a performance management model for SPI along with specific advice for managers and developers in software organizations that want their organization to excel.
Readability:	
8. Use clear, simple, and concise style in the write-up.	Partial. Since a doctoral dissertation is mainly read by the research community, the chosen style is deliberately academic. However, the discussion of results, and particularly the implications for practice (*Section 9.6*), is more "managerial" in nature, following magazine style rather than journal style.
Editorial process:	
9. Set the goal of publishing manuscripts as being *both* rigorous and relevant	Yes, this has been clearly stated throughout the thesis, and in the present evaluation.

Thus, based on the evaluation of methodological rigor presented in *Table 10.1* and the evaluation of relevance to practice presented in *Table 10.2*, we conclude that *the present investigation can be considered both rigorous and relevant*.

10.4 Limitations

Although we have discussed several implications of our results for the theory, methodology, and practice of SPI, the research reported here is not without its limitations. First, the model building part of the investigation rests in the interpretive research tradition, which holds that reality is a subjective construction of the mind, and that its form and content depends on the individual holding the construction. Even though we applied grounded theory and the fundamental principle of the hermeneutic circle (Klein and Myers, 1999), which was explicitly built into the dynamic model of SPI through the local and organizational levels, the model is still a subjective construction. Besides, the process of interpretation is iterative, which means that the subjective constructions and their associated "realities" are alterable. Therefore, we cannot claim that the model reflects objective knowledge (but then again, which model can?). The question, therefore, is not whether the model is more or less "true", in any absolute sense, but simply if it is more or less informed and/or sophisticated (Guba and Lincoln, 1994).

Second, the results of the hypothesis-testing part of this research have been discussed as though the facilitating factors caused SPI success. This part of the investigation was, however, cross-sectional, and these assumptions of causality are not technically justified. It is possible, for example, that there are complex feedback mechanisms by which performance in one time period is affected by performance in previous periods (March and Sutton, 1997). Both success and lack of success might very well be self-reinforcing. On the other hand, there could also be negative feedback mechanisms by which SPI success or failure creates countervailing tendencies. It is possible, therefore, that performance below target levels increases organizational efforts and, thus, the likelihood of subsequent success. Accordingly, success can also be seen as a trigger for adjustments in the opposite direction through decreased innovation or increased slack. Therefore, many of the facilitating factors that seem likely to influence SPI success can themselves be influenced by prior success or lack of success. Finding the "true" causal structure of SPI success based on incomplete information generated by prior experience is, therefore, problematic.

Third, although generally accepted psychometric principles were used to develop the measurement instrument, the variables were still measured on the basis of subjective performance definitions. Performance measures such as the return on investment, net present value, and payback periods are often regarded as objective measures. However, attempts to provide objective definitions of such measures may be as open to criticism as subjective definitions (Berry and Jeffery, 2000; Saarinen, 1996). This points to a general problem of defining and measuring success in studies of SPI. The question, therefore, is not whether such measures are subjective or not, but what purpose they serve.

Finally, a further complication is that the independent variables were assessed using retrospective recall. This involves a risk for the introduction of retrospective bias (Fischhoff, 1975). It is possible, therefore, that performance information itself colors subjective memories and perceptions of possible causes of SPI success or failure. Software organizations are constantly worried about their performance, and "common wisdom" has many explanations for good and poor performance. As a result, retrospective reports of independent variables may be less influenced by memory than by a reconstruction that connects common wisdom with the awareness of performance results (March and Sutton, 1997).

Despite these limitations, this study contributes to the growing literature on ESE research and provides empirical support for the importance of organizational issues in SPI.

10.5 Further Research

The results of this thesis, its contributions and limitations, point to several possibilities for further research, with relevance for theory, methodology, and practice:

- **Theory of organizational learning and change in software organizations**. A more detailed theory of organizational learning and change in software organizations should be developed to enhance the dynamic model of SPI, and to better understand the concept of the learning software organization. Such a theory should be built on the basis of *in situ* fieldwork. That is, the researcher should actively participate in situation specific practices in a few organizations in order to get "rich" data and an in-depth understanding of the central elements in the theory. On the basis of such cases, a possible approach would be to use qualitative techniques such as grounded theory in the actual theory building process.
- **SPI success measurement**. SPI success measurement is a controversial issue and more research is needed to study it. Several levels of analysis are possible – e.g. individual, group, process, and organization – each with complex interactions with the others. Also, several, and possibly conflicting, dimensions (e.g. faster, better, and cheaper) and viewpoints (e.g. economic, engineering, organizational, and socio-technical) are relevant. Research should be related to the study of new and improved measures of SPI success, comparison of measurement instruments, and validation of SPI success measures, using both positivist and interpretive approaches.
- **The role of context variables in SPI**. Context variables, such as organizational size and environmental turbulence, did not play an important role in *predicting* SPI success in this study. However, they played an important role in determining the *approach* taken to SPI. Also, under other circumstances, the independent variables of the present study could act as moderating or mediating variables in other studies. Research is needed to investigate the importance of such variables to several types of SPI problems and to validate the approaches proposed for solving them. Such studies should combine factor research with process research in order to provide satisfactory levels of external validity.
- **The nature of causality in SPI studies**. Laboratory experiments regarding the role of retrospective bias in assessing independent variables based on retrospective recall should be undertaken to study the nature of causality in SPI. Such experiments should subsequently be extended to field situations in order to study the "true" nature of such causality. The research design for such studies should be based on a pretest-posttest control group design, preferably with professional software developers and managers in both laboratory and field settings.
- **Diversity of ESE research approaches**. Efforts should be made to develop a diversity of ESE research approaches, suited for different purposes. Of specific concern, however, are research approaches that are able to support the development and testing of theories to improve our *understanding* of practice, while simultaneously supporting the development and testing of concepts to *support* industrial practice. Research is needed to develop action research as an approach to ESE research that satisfies the requirements of scientific rigor, while at the same time focusing on relevance for practice. Such approaches would make it easier to combine the role of the researcher with the role of the consultant in SPI.

- **Guidelines for SPI in small organizations**. Practical guidelines for process definition, assessment, improvement, and innovation in small organizations should be developed. Such approaches should combine knowledge creation and socio-technical concepts such as "co-determination", "minimum critical specification", and "autonomous workgroups" with improvisation and living organizational memories. Of particular importance, here, is work that is being done to study the relevance of improvisation in jazz and team games to organization development in order to e.g. reduce the time between planning and action (see Dybå, 2000b). Eventually, the guidelines should be evaluated in an industrial context through the use of field experiments, surveys, case studies, or action research.

There are certainly other directions for further research. However, the value of any such future work depends on the specific goals of each particular investigation.

10.6 Final Remarks

The results of the empirical investigations presented in this thesis support our claim that organizational issues are as important in SPI as technology, if not more so. Therefore, like Kraut and Streeter (1995), who studied coordination in software development, we believe that the large literature on organizational theory, developed in non-software settings, has more relevance to software development than has previously been recognized in software engineering.

Finally, and in accordance with our constructivist standpoint, we could not say, "Here's how to do it". Rather, we believe that each software organization must use its own learning capacity to invent the specific processes and enabling factors that it needs to succeed with organizational learning and SPI. Nevertheless, we believe that the results of this investigation can provide valuable inputs to such organization specific processes and that learning from others can be a valuable resource.

Appendix A: Terminology

A.1 Abbreviations

ACM	Association for Computing Machinery
ami	**a**pplication of **m**etrics in **i**ndustry
BPR	Business Process Reengineering
BSC	Balanced Scorecard
CBA IPI	CMM Based Appraisal for Internal Process Improvement
CEO	Chief Executive Officer
CMM	Capability Maturity Model
CMMI	CMM Integration
EASE	Empirical Assessment in Software Engineering
EF	Experience Factory
EFQM	European Foundation for Quality Management
ESE	Empirical Software Engineering
ESSI	European Systems and Software Initiative
GQM	Goal Question Metric
GSFC	Goddard Space Flight Center
IDEAL	Initiating-Diagnosing-Establishing-Acting-Learning (model)
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IS	Information Systems
ISERN	International Software Engineering Research Network
ISO	International Organization for Standardization
KPA	Key Process Area
LO	Learning Organization
LSO	Learning Software Organization

MIS	Management Information Systems
NASA	National Aeronautics and Space Administration
NTNU	The Norwegian University of Science and Technology
OD	Organization Development
OL	Organizational Learning
OMIS	Organizational Memory Information System
PDCA	Plan-Do-Check-Act (cycle)
PIE	Process Improvement Experiment
PROFIT	Process Improvement for IT-Industry (project)
QIP	Quality Improvement Paradigm
R&D	Research and Development
ROI	Return on Investment
SCAMPI	Standard CMMI Assessment Method for Process Improvement
SD	(1) Software Development, (2) Standard Deviation
SEI	Software Engineering Institute
SEL	Software Engineering Laboratory
SEPG	Software Engineering Process Group
SINTEF	The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology
SME	Small and Medium-sized Enterprises
SPA	Software Process Assessment
SPC	Statistical Process Control
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability dEtermination.
SPIQ	Software Process Improvement for better Quality (project)
STS	Socio-Technical System
SW	Software
SW-CMM	Capability Maturity Model for Software
TELMET	Telecommunication Metrics Approach (project)
TQM	Total Quality Management
UiO	University of Oslo

A.2 Definitions

Action Research: An iterative process involving researchers and practitioners acting together on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning (Avison *et al*., 1999).

Benchmarking: An ongoing investigation and learning experience that ensures that best practices are uncovered, analyzed, adopted, and implemented.

Business orientation: The extent to which SPI goals and actions are aligned with explicit and implicit business goals and strategies.

Case study: An empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident (Yin, 1994).

Conceptualization: The process of taking an abstract construct and refining it by giving it a conceptual or theoretical definition (Neuman, 2000).

Concern for measurement: The extent to which the software organization collects and utilizes quality data to guide and assess the effects of SPI activities.

Construct validity: The degree to which a measurement instrument represents and acts like the processes being measured (Cronbach, 1971).

Content validity: The degree to which the items in a measurement instrument represent the domain or universe of the processes under study (Cronbach, 1971).

Credibility: The arguments and the processes necessary for having someone trust research results (Greenwood & Levin, 1998).

Criterion validity: The degree to which a measurement instrument is able to predict a variable that is designated a criterion (Anastasi & Urbina, 1997).

Cross-sectional research: Research in which the measurements on the variables of interest are taken at one point in time.

Customer satisfaction: The difference between the customers' perceived performance and their needs and expectations (Lynch & Cross, 1991).

Data: Symbols, e.g. numbers or letters.

Deductive research: Research that begins with an abstract, logical relationship among concepts, formulated as a hypothesis, and then moves toward concrete empirical evidence by testing the hypothesis.

Dependent variable: The variable in a study that is the effect or outcome of the independent variable. It "depends on" the cause (Davis, 1996).

Deuterolearning: Learning how to learn (Bateson, 1972).

Double-loop learning: Learning that results in a change in the values of theory-in-use, as well as in its strategies and assumptions (Argyris & Schön, 1996).

Employee participation: The extent to which employees use their knowledge and experience to decide, act, and take responsibility for SPI.

Epistemology: Questions regarding the nature of the relationship between the researcher and what can be known (Guba & Lincoln, 1994).

ESE research: Analysis based on the investigation of actual practice for the purpose of discovering the unknown or testing a hypothesis.

Espoused theory: The theory of action, which is advanced to explain or justify a given pattern of activity (Argyris & Schön, 1996).

Experience: (1) Direct observation of or participation in events as a basis of knowledge. (2) The fact or state of having been affected by or gained knowledge through direct observation or participation (Merriam-Webster, 1998).

Experience base: An integrated collection of experience packages.

Experience Factory: An organization that supports reuse of experience and collective learning within a software organization (Basili *et al*., 1994a).

Experience package: Externalized experience, packaged in a form that makes it available for future use.

Experiment: An empirical inquiry that investigates explanatory relations. It is especially well suited for answering questions about how and why (Yin, 1994).

Externalization: The enactment of local reality; the process of creating organizational institutions, systems, and routines by taking action and using language to express oneself to the external world (Berger & Luckmann, 1966).

Facilitating factors: The conditions that facilitate or enable knowledge creation and SPI. They are the key factors for success that the software organization must put in place in order to facilitate the organization's learning cycle and improve its development processes.

Fact: Those things or phenomena that we believe are true.

Feedback session: A learning forum that is regularly held during a measurement program.

Goal-Question-Metric: An approach to goal-oriented measurement in software projects (Basili *et al*., 1994b).

Hypothesis: A conjectural statement of the relationship between two or more variables that carry clear implications for testing the stated relation (Kerlinger, 1986).

Improvisation: The reworking of precomposed material and designs in relation to unanticipated ideas conceived, shaped, and transformed under the special conditions of performance, thereby adding unique features to every creation (Berliner, 1994).

Independent variable: The variable in a study that is the presumed cause of the presumed effect. It is "independent of" prior causes that act on it (Davis, 1996).

Inductive research: Research that begins with detailed observations of the world, moving toward abstract generalizations and theories, and eventually to the formulation of hypotheses.

Information: (1) Externalized knowledge. (2) A reduction in uncertainty (Shannon & Weaver, 1949) or the differences that make a difference (Bateson, 1979).

Instrument construction: The process of developing the data collection device in order to define and obtain relevant data for a given research question.

Internalization: The process by which individuals or groups of individuals give meaning to organizational reality, making it part of their own local reality (Berger & Luckmann, 1966).

Interpretivism: A philosophy that maintains that reality is a subjective construction of the mind, and that its form and content depends on the individual holding the construction (Lee, 1991).

Intervening variable: A variable that emerges as a function of the independent variable operating in a situation and helps to explain the influence of the independent variable on the dependent variable (Davis, 1996).

Involved leadership: The extent to which leaders at all levels in the organization are genuinely committed to and actively participate in SPI.

Knowing: The epistemological dimension of action (Cook & Brown, 1999).

Knowledge: (1) The fact or condition of knowing something with familiarity gained through experience or association (Merriam-Webster, 1998). (2) Justified true belief.

Knowledge creation: A continuous and dynamic interaction between tacit and explicit knowledge (Nonaka & Takeuchi, 1995).

Leadership: See 'Involved leadership'.

Learning: The process whereby knowledge is created through the transformation of experience (Kolb, 1984).

Learning cycle: A dialectic process that integrates local experience and organizational concepts.

Learning software organization: A software organization that promotes improved actions through better knowledge and understanding.

Learning strategy: The extent to which a software organization is engaged in the exploitation of existing knowledge and in the exploration of new knowledge.

Lessons learned: Concise and insightful experiences that can be fruitfully applied in future endeavors.

Local reality: The way an individual or a group of individuals perceive the world in which they live (Berger & Luckmann, 1966).

Longitudinal research: Research in which measures of the same sample or population are taken repeatedly (at least twice) through time.

Measure: Data assigned to an entity in order to characterize an attribute (Fenton & Pfleeger, 1996).

Measurement: The process by which data is assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules (Fenton & Pfleeger, 1996).

Measurement instrument: A data collection device (in this study a questionnaire).

Measurement scale: A device that is used to assign numbers to aspects of objects and events.

Method: A procedure or process for attaining an object (Merriam-Webster, 1998).

Methodology: Questions regarding how the researcher gains knowledge about the phenomenon under study (Guba & Lincoln, 1994).

Metric: A property of a process, product, or resource.

Model: Abstractions of reality, allowing us to strip away detail and view an entity or concept from a particular perspective (Fenton & Pfleeger, 1996).

Moderating variable: A variable that has a strong effect on an independent-dependent relationship (Davis, 1996).

Observation: Perception of reality by noting some object or the occurrence of some phenomenon in the immediate environment (Davis, 1996).

Ontology: Questions regarding the form and nature of reality and, therefore, about what there is that can be known about it (Guba & Lincoln, 1994).

Operationalization: The process of linking a conceptual definition to a specific set of measurement techniques or procedures (Neuman, 2000).

Organizational context: The general environment that imposes constraints and opportunities about what the organization can and cannot do with respect to organizational learning and SPI.

Organizational issues: Any distinct area on the interface between the technical system and the social system, which can enable software process improvements. The thesis focuses on two classes of such issues: (1) the socially constructed learning processes, and (2) the key factors that facilitate improvements in organizational performance.

Organizational learning: The process of improving actions through better knowledge and understanding (Fiol & Lyles, 1985). For alternative definitions, see Figure 4.2.

Organizational memory: The means by which a software organization's knowledge from the past is brought to bear on present activities.

Organizational reality: The social order or institution that exists because everybody relate to it in their actions (Berger & Luckmann, 1966).

Paradigm: A commonality of perspective, which binds the work of a group of theorists together (Burrell and Morgan, 1979).

Population: The complete set of units of analysis that are under investigation (Davis, 1996).

Positivism: A philosophy that argues for the application of the methods of the natural sciences to the social sciences and thereby presupposes the unity of science (Delanty, 1997).

Pragmatism: A philosophy based on the proposition that researchers should use whatever philosophical and/or methodological approach that works for the particular research problem under study (Tashakkori & Teddlie, 1998).

Process: A partially ordered set of activities to produce a specific output.

Process assessment: An appraisal or review of a (software) organization and its processes to advice its management and professionals on how they can improve their operation (Humphrey, 1989).

Process capability: The ability of a process to achieve a required goal (ISO/IEC TR 15504-9: 1998).

Process conformance: The degree of agreement between a process execution and a process model (Sørumgård, 1997).

Process improvement: An intervention strategy that involves continuous or incremental change to the organization and its processes.

Process innovation: An intervention strategy that involves radical change to the organization with replacement of its processes.

Process model: An abstract description of a (software) process.

Quality: (1) The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs (ISO 8402: 1994). (2) Ability of a set of inherent characteristics of a product, system, or process to fulfill requirements[1] of customers and other interested parties (ISO 9000:2000).

Quality assurance: Part of quality management, focused on providing confidence that quality requirements are fulfilled (ISO 9000:2000).

Quality Improvement Paradigm: A goal-driven framework for continuous improvement of software development and maintenance (Basili *et al*., 1994a).

Quality management: Coordinated activities to direct and control an organization with regard to quality (ISO 9000:2000).

Quality management system: System to establish a quality policy and quality objectives and to achieve those objectives (ISO 9000:2000).

Questionnaire item: A question and its associated measurement scale.

Reliability: The consistency and stability of a score from a measurement scale (Anastasi & Urbina, 1997).

Research design: A guide used by researchers in their quest to answer the research question and solve the problems under study; a "blueprint" of research (Yin, 1994).

Sample: The subset of units of analysis chosen for study from a population (Davis, 1996).

Sampling frame: A physical representation of objects, individuals, groups, etc., important to the development of the final study sample (Davis, 1996).

Single-loop learning: Instrumental learning that changes strategies of action or assumptions underlying strategies in ways that leave the values of a theory of action unchanged (Argyris & Schön, 1996).

[1] In this context, requirement is defined as "need or expectation that is stated, customarily implied or obligatory" (ISO 9000:2000).

Social constructivism: A philosophy that maintains that social reality is constructed through a dialectical process between externalization of individual subjective realities, through action, resulting in objective reality; and the internalization of this objective reality, through sensemaking, into subjective reality (Berger & Luckmann, 1966).

Software: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system (IEEE Std 610.12-1990).

Software development process: The process by which user needs are translated into a software product (IEEE Std 610.12-1990).

Software engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1) (IEEE Std 610.12-1990).

Software organization: A whole company or an independent business unit inside a larger company that has software development as its primary business.

Software process: A coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product (Fuggetta, 2000).

Software process improvement: Activities to (1) define and model a software process, (2) to assess the process, (3) to refine the process, and (4) to innovate a new process.

Software product: (1) The complete set of computer programs, procedures, and possibly associated documentation and data designated for delivery to a user. (2) Any of the individual items in (1) (IEEE Std 610.12-1990).

SPI success: The performance or results of the organization's SPI activities. This is the dependent variable that is used to measure that gains have in fact been made with respect to organizational behavior and performance.

Survey: A retrospective study of a situation that investigates relationships and outcomes. It is especially well suited for answering questions about what, how much, and how many as well as questions about how and why (Pinsonneault & Kraemer, 1993).

Techniques: Technical and managerial procedures that aid in the evaluation and improvement of the software development process (IEEE Std 983-1986).

Technology: (1) Physical objects or artifacts, (2) activities or processes, and (3) knowledge or potential actions (MacKenzie & Wajcman, 1985).

Theory: An interrelated set of statements of relationships whose purpose is to explain and predict (Davis, 1996).

Theory-in-use: The theory of action, which is implicit in the performance of a pattern of activity (Argyris & Schön, 1996).

Validity: (1) The best available approximation to the truth of propositions, including propositions about cause (Cook & Campbell, 1979). (2) A measurement scale is valid if it does what it is supposed to do and measures what it is supposed to measure (Cronbach, 1971).

Variable: A symbol or a concept that can assume any one of a set of values (Davis, 1996).

Appendix B: Interview Guide

Q1. Knowledge of routines.

Q1.1. Describe the (possible) contents in routines being used for software development.

Q1.2. How were these routines introduced in the company?

Q1.3. What was the purpose of these routines?

Q1.4. What is you personal impression of the routines?

Q2. Use of routines.

Q2.1. What status does a given routine have among developers?

Q2.2. To what degree are the routines actually being used?

Q2.3. Who are the most active/passive users?

Q2.4. What is the availability of the routines?

Q2.5. How is follow-up and control of usage done?

Q3. Updating of routines.

Q3.1. What procedures are used to update the routines?

Q3.2. Who participates in the update activities?

Q4. Routines as a medium for transfer of knowledge and experience.

Q4.1. Do you regard written routines as an efficient medium for transfer of knowledge and experience?

Q4.2. What alternatives to written routines do you think are useful or in use in the company?

Q4.3. What barriers against transfer of experiences do you think are most important?

Appendix C: Example of Results from an Exploratory Study in CX[2]

In this appendix, we include the results from an exploratory study that was conducted in Company X to validate the literature-derived factors enabling SPI success. The data collection method used to derive key factors for success was three questionnaire items and a follow-up group interview (feedback session). For the purpose of the results presented in this chapter, data collection focused on answering one question:

- *In your opinion, what are the three most important factors enabling SPI success in your organization?*

In order to gain more background information about the enabling factors of SPI success, our questioning strategy included two additional questions. First we asked the subjects about their most important argument *in favor* of SPI in their organization. Second we asked the subjects about their most important argument *against* SPI in their organization. The answers to these questions gave us valuable information for interpreting the enabling factors.

Ten process managers answered the three questions and participated in the subsequent feedback session. The result of coding the field data into theoretical concepts was the set of five enabling factors for SPI success shown in *Table C.1*. Each factor was mentioned by at least three subjects.

Answers to the questions:

1. **What is the most important argument *in favor* of SPI in your organization?**

- Bedre muligheter for ”bedre” prosjekter, dvs. de som oppfattes som ønskelige prosjekter som ”vi” har lyst til å jobbe med.
- Øke profesjonaliteten innad og høyne posisjonen mht. kvalitet og effektivitet utad.
- Investering i mer effektiv produksjon. Bedre inntjening, høyere salg. Mer tilfredsstillende arbeidssituasjon.
- Avgjørende for fremtidig konkurransedyktighet.

[2] This study was one of four such studies performed in four different companies with a total of 54 managers, developers, and customer representatives to validate the literature-derived factors.

- Flere får nytte til samme kostnad ved vekst i antall ansatte.
- Vi må forbedre prosessene for å bevare og øke vår konkurransekraft.
- Gjennom prosessforbedring vil vi kunne levere produkter med høyere kvalitet og til lavere kostnad. Dette vil kunne påvirke arbeidstilfredsheten i positiv retning.
- Tilpasning til stadig forbedret forståelse av egen organisasjon. Tilpasning til eksterne rammebetingelser. Ny teknologi, nye leveransekanaler, nye leveransemuligheter.
- Kontinuerlig tilpasning til ekstern forandring. Dynamisk og utfordrende arbeidsplass.
- Det er og vil alltid være, mye som kan og bør forbedres.

2. What is the most important argument *against* of SPI in your organization?

- Vil skape ”kultur”-konflikter i organisasjonen.
- Fare for å dempe den enkeltes kreativitet og ansvarlighet overfor sine oppgaver ved å innføre for rigide systemer rundt den enkeltes arbeidsoppgaver.
- Vanskelig å gjennomføre pga. liten tradisjon i styring/byråkratisering. Enkelte kan være lite villige til å ”se seg selv i speilet”. Større arbeidsbelastning på hver medarbeider.
- Innebærer mye innsats som kanskje ikke betaler seg tilbake, f.eks. i form av å oppnå forståelse for dette i hele organisasjonen.
- Koster mye overbevisningskraft.
- Ingen gode argumenter mot prosessforbedring.
- Dersom prosessforbedring bare betyr formalisering av rutiner vil dette forvitre den enkeltes ansvarlighet.
- Det er tross alt ressurskrevende.
- Det tar tid, ressurser og fokus vekk fra prosjekter som paradoksalt nok ellers kunne levere et bedre produkt til kunden (på kort sikt?).
- Viktig hele tiden å passe på at ikke selve arbeidet med prosessforbedring stjeler fokus fra det som skal forbedres.

3. In your opinion, what are the three most important factors enabling SPI success in your organization?

- Ledelsesprioritering. Opplæring av medarbeidere. Hensiktsmessige forandringer.
- Enkle, oversiktlige rutiner fra basistrinnene i tilbuds- og prosjektarbeid. Bedre prosjektplanlegging og –styring. Klare retningslinjer for QA-gjennomgang.
- Opplæring og motivasjon. Felles forståelse av mål. Må kunne se gevinster.
- Formalisering av estimering, inspeksjon og måling slik at dette støtter opp om overordnede mål. Prosjektgjennomføringsstøtte inkl. forbedringsmomenter. Innføring av obligatoriske standarder.
- Iherdighet og kontinuitet blant de som skal følge opp forbedringsarbeidet og rutinene rundt dette. Aksept blant de som prioriterer personellressurser. Anerkjennelse av evt. ekstra tidsbruk til prosessforbedring på kort sikt for å få fordeler på lang sikt.

- Kraftigere, mer energisk implementering av allerede etablerte prosesser. Bedre synliggjøring og bedre oppfølging av selve forbedringsarbeidet. Bedre verktøy for å måle og etterprøve effekten av forbedringstiltak.
- Utføres i konkrete prosjekter. Tas inn som en del av prosjektplanene. Oppfølging i organisasjonen på hva som oppnås.
- Omforent situasjonsforståelse (intern og ekstern). Medvirkning. Prioritering.
- Villighet til å prøve og feile. Kunnskapsdeling. Oversikt over faktiske forhold.
- Toppledelseforankring. Oppleves meningsfylt/nyttig av de ansatte. Forbedre bottom-line resultat.

Table C.1: Factors enabling SPI success in Company X.

Enabling Factors	**Explanation of Enabling Factors**
Leadership Involvement	Ledelsesprioritering. Toppledelseforankring. Aksept blant de som prioriterer personellressurser. Prioritering. Anerkjennelse av tidsbruk til prosessforbedring på kort sikt for å få fordeler på lang sikt.
Employee Participation	Medvirkning. Opplæring av medarbeidere. Opplæring og motivasjon. Oppleves meningsfylt/nyttig av de ansatte.
Learning from Experience	Kunnskapsdeling. Hensiktsmessige forandringer. Enkle, oversiktlige rutiner. Formalisering. Innføring av standarder.
Concern for Measurement	Måle og etterprøve effekten av forbedringstiltak. Synliggjøring og oppfølging av forbedringsarbeidet. Oversikt over faktiske forhold. Må kunne se gevinster. Oppfølging i organisasjonen på hva som oppnås.
Business Orientation	Forbedre bottom-line resultat. Omforent situasjonsforståelse (intern og ekstern). Felles forståelse av mål. Utføres i konkrete prosjekter. Tas inn som en del av prosjektplanene. Prosjekt-gjennomføringsstøtte.

Appendix D: Original List of Facilitating Factors and Indicators

The items marked by an asterisk (*) were dropped after the expert review. Items marked by two asterisks (**) were eventually deleted in order to improve the reliability of the final instrument.

Key Factors	**Potential Indicators**
Business Orientation	• amount of specific goals for SPI within the company* • extent of goal-orientation in SPI activities • extent of long term planning of SPI activities* • extent to which SPI goals and policy are understood • degree of integrating SPI actions with "real" work • extent to which SPI goals are aligned with business goals • thoroughness in the priority of improvement actions* • degree of balance between short-term and long-term goals
Leadership Involvement	• degree of participation by management in the SPI process • acceptance of responsibility for SPI by management • extent to which top management has objectives for software performance • degree to which top management is evaluated for software performance* • degree to which management considers SPI as a way to increase competitive advantage • amount of review of SPI issues in top management meetings
Concern for Measurement	• importance of measuring performance • availability of quality data (defects, timeliness, etc) • extent to which quality data are available to developers • extent to which quality data are available to managers • extent to which quality data, charts, etc. are displayed in corridors etc* • extent to which simple statistical tools are used* • amount of feedback provided to developers on their performance

Key Factors	Potential Indicators
Employee Participation	• extent of employee involvement in decisions about what should best be done at their own level (co-determination) • extent of employee participation in development planning** • extent of developer participation in the formalization of routines • extent to which employees contribute with improvement proposals • extent of employee interaction with customers and suppliers* • extent to which employees have responsibility for SPI • amount of training in SPI concepts* • extent of developer participation in goal-setting, data-analysis, and interpretation • extent of participatory management* • extent of on-going dialogue and discussion about software development • extent of on-going dialogue and discussion about SPI
Exploitation of existing knowledge	• extent to which existing knowledge is exploited • extent of learning from past experience • degree of elaboration of existing ideas, strategies, and technologies* • degree of systemization of past experience • importance of formulating experience into explicit concepts* • degree to which formal routines are based on past experience • extent to which computerized organizational memory are used* • importance of narrow (specialized) skills* • importance of maximum breakdown of task* • extent of searching for stability* • degree of internal experience transfer • extent of risk aversion**
Experimentation	• extent of exploration of new knowledge* • degree of experimentation with new ideas, strategies, and technologies • importance of exploring opportunities* • extent to which innovation/change is encouraged • degree of learning from other organizations* • extent to which risk taking is encouraged* • importance of multiple broad skills* • degree of adaptability to rapid change, increasing complexity and environmental uncertainty • ability to question underlying values
Operational variety	• importance of matching the variety and complexity of the organization's environment • extent of variety in the use of methods, techniques and tools • extent to which diversity is appreciated* • degree of detail in task specifications (minimum critical specification) • degree of flexibility in task execution • importance of optimum variety in task performance* • importance of optimum grouping of tasks*

Appendix E: Measurement Instrument[3]

Participants in the study were asked to mark or circle the best response to each statement on the following 5-point scale: "Strongly disagree", "Disagree", "Neither agree nor disagree", "Agree", and "Strongly agree" for each of the following factors.

Business Orientation

1. We have established unambiguous goals for the organization's SPI activities.
2. There is a broad understanding of SPI goals and policy within our organization.
3. Our SPI activities are closely integrated with software development activities.
4. Our SPI goals are closely aligned with the organization's business goals.
5. We have a fine balance between short-term and long-term SPI goals.

Leadership Involvement

6. Management is actively supporting SPI activities.
7. Management accepts responsibility for SPI.
8. Management considers SPI as a way to increase competitive advantage.
9. Management is actively participating in SPI activities.
10. SPI issues are often discussed in top management meetings.

Employee Participation

11. Software developers are involved to a great extent in decisions about the implementation of their own work.
12. Software developers are actively contributing with SPI proposals.
13. Software developers are actively involved in creating routines and procedures for software development.
14. We have an on-going dialogue and discussion about software development.
15. Software developers have responsibility related to the organization's SPI activities.

[3] Starred items were removed from the final version of the instrument and should not be used.

16. Software developers are actively involved in setting goals for our SPI activities.
17. We have an on-going dialogue and discussion about SPI.

Concern for Measurement

18. We consider it as important to measure organizational performance.
19. We regularly collect quality data (e.g. defects, timeliness) from our projects.
20. Information on quality data is readily available to software developers.
21. Information on quality data is readily available to management.
22. We use quality data as a basis for SPI.
23. Our software projects get regular feedback on their performance.

Exploitation of Existing Knowledge

24. We exploit the existing organizational knowledge to the utmost extent.
25. We are systematically learning from the experience of prior projects.
26. Our routines for software development are based on experience from prior projects.
27. We collect and classify experience from prior projects.
28. We put great emphasis on internal transfer of positive and negative experience.
29. To the extent we can avoid it, we do not take risks by experimenting with new ways of working.*

Exploration of New Knowledge

30. We are very capable at managing uncertainty in the organization’s environment.
31. In our organization, we encourage innovation and creativity.
32. We often carry out trials with new software engineering methods and tools.
33. We often conduct experiments with new ways of working with software development.
34. We have the ability to question “established” truths.
35. We are very flexible in the way we carry out our work.
36. We do not specify work processes more than what is absolutely necessary.
37. We make the most of the diversity in the developer’s skills and interests to manage the variety and complexity of the organization’s environment.

Organizational Performance

1. Our SPI work has substantially increased our software engineering competence.
2. Our SPI work has substantially improved our overall performance.
3. Over the past 3 years, we have greatly reduced the cost of software development.
4. Over the past 3 years, we have greatly reduced the cycle time of software development
5. Over the past 3 years, we have greatly increased our customer’s satisfaction.

Appendix F: Questionnaire

Spørreundersøkelse om

PROSESSFORBEDRING I PROGRAMVAREUTVIKLING

Innledning

I samarbeid med Norges forskningsråd gjennomfører Norges teknisk-naturvitenskaplige universitet (NTNU) og SINTEF en spørreundersøkelse om forbedringsarbeid knyttet til utvikling av programvare. Denne undersøkelsen er del av en større nasjonal satsing på kunnskapsformidling og forbedring i norske programvaremiljøer. Resultatene fra spørreundersøkelsen inngår også som en del av et doktorgradsarbeid ved NTNU.

Hensikten med undersøkelsen er å studere de organisasjonsmessige forholdene som ofte legger føringer for endrings- og forbedringsarbeid i teknologibedrifter. I spørreskjemaet er forbedringsarbeid definert som alt organisasjonsmessig og teknisk rettet arbeid for å forbedre eksisterende måter å utvikle programvare på. Hoveddelen av spørreskjemaet består av en rekke utsagn som karakteriserer en organisasjon. Vi ber om at du, for hvert utsagn, angir i hvor stor grad du er enig/uenig i at dette utsagnet karakteriserer *din* organisasjon. Hvis det er flere enheter i organisasjonen som utvikler programvare, ber vi om at du svarer for den enheten du kjenner best.

Din deltakelse i undersøkelsen er viktig fordi du har en sentral posisjon i en bedrift som utvikler programvare, og fordi du har viktige erfaringer i forhold til hva som skal til for å lykkes med forbedringsarbeid i en kunnskapsbedrift. Utfylling av det vedlagte spørreskjemaet tar ca 10 minutter.

Alle opplysninger vil bli behandlet fortrolig. Hverken navn på personer eller bedrifter vil bli nevnt i forskningsrapporter. Hvis du ønsker å få tilsendt rapporten fra denne undersøkelsen, vennligst skriv navn og adresse nederst på denne siden eller send oss et visittkort. Eventuelle spørsmål om undersøkelsen kan rettes til undertegnede via e-post: Tore.Dyba@informatics.sintef.no eller på telefon 73 59 29 47.

Jeg vil sette stor pris på om du kan ta deg tid til å fylle ut spørreskjemaet og returnere det i vedlagte svarkonvolutt i løpet av to uker. På forhånd: tusen takk for hjelpen!

Med vennlig hilsen

Tore Dybå
SINTEF Tele og data
7465 Trondheim

Vennligst marker hvor godt du mener følgende utsagn beskriver forbedringsarbeid i *din* organisasjon.

Læring fra egne erfaringer:	**Sterkt uenig**	**Uenig**	**Hverken enig eller uenig**	**Enig**	**Sterkt Enig**
1. Vi utnytter den eksisterende kunnskapen i bedriften maksimalt.	☐	☐	☐	☐	☐
2. Vi lærer systematisk fra egne prosjekterfaringer.	☐	☐	☐	☐	☐
3. Utviklingsrutinene våre er basert på erfaringer fra tidligere prosjekter.	☐	☐	☐	☐	☐
4. Vi tar vare på og systematiserer erfaringer fra gjennomførte prosjekter.	☐	☐	☐	☐	☐
5. Vi legger stor vekt på intern spredning av positive og negative erfaringer.	☐	☐	☐	☐	☐
6. I den grad vi kan unngå det, tar vi ikke risiko ved å prøve ut arbeidsmetoder som vi ikke har erfaring med.	☐	☐	☐	☐	☐

Ledelsesengasjement:	**Sterkt uenig**	**Uenig**	**Hverken enig eller uenig**	**Enig**	**Sterkt Enig**
7. Ledelsen gir aktiv støtte til forbedringsarbeid	☐	☐	☐	☐	☐
8. Ledelsen tar ansvar for forbedringsarbeid.	☐	☐	☐	☐	☐
9. Ledelsen innser at forbedringsarbeid kan bidra til økt konkurransekraft.	☐	☐	☐	☐	☐
10. Ledelsen deltar aktivt i bedriftens forbedringsarbeid.	☐	☐	☐	☐	☐
11. Forbedringsarbeid blir ofte diskutert i ledermøter.	☐	☐	☐	☐	☐

Forretningsorientering:	**Sterkt uenig**	**Uenig**	**Hverken enig eller uenig**	**Enig**	**Sterkt Enig**
12. Vi har etablert klare mål for bedriftens forbedringsarbeid.	☐	☐	☐	☐	☐
13. Det er bred forståelse for forbedringsmål og –policy i bedriften.	☐	☐	☐	☐	☐
14. Forbedringsarbeidet er tett integrert med utviklingsarbeid.	☐	☐	☐	☐	☐
15. Forbedringsarbeidet bygger opp under bedriftens overordnede forretningsmessige mål.	☐	☐	☐	☐	☐
16. Vi har en god balanse mellom kortsiktige og langsiktige forbedringsmål.	☐	☐	☐	☐	☐

Måling:	**Sterkt uenig**	**Uenig**	**Hverken enig eller uenig**	**Enig**	**Sterkt Enig**
17. Vi anser det som viktig å måle bedriftens prestasjoner.	☐	☐	☐	☐	☐
18. Vi samler regelmessig inn kvalitetsdata (feil, forsinkelser, endringer etc.) fra prosjektene våre.	☐	☐	☐	☐	☐
19. Kvalitetsdata er lett tilgjengelig for utviklere.	☐	☐	☐	☐	☐
20. Kvalitetsdata er lett tilgjengelig for ledelsen.	☐	☐	☐	☐	☐
21. Kvalitetsdata benyttes som grunnlag for forbedring.	☐	☐	☐	☐	☐
22. Utviklingsprosjektene får regelmessig tilbakemelding (feedback) på det arbeidet de utfører.	☐	☐	☐	☐	☐

Vennligst marker hvor godt du mener følgende utsagn beskriver forbedringsarbeid i *din* organisasjon.

Utprøving og fleksibilitet:	**Sterkt uenig**	**Uenig**	**Hverken enig eller uenig**	**Enig**	**Sterkt Enig**
23. Vi er svært dyktige til å håndtere usikkerhet i bedriftens omgivelser.	☐	☐	☐	☐	☐
24. I vår bedrift oppfordres det til nytenking og kreativitet.	☐	☐	☐	☐	☐
25. Vi prøver ofte ut nye metoder og verktøy.	☐	☐	☐	☐	☐
26. Vi eksperimenterer ofte med nye måter å arbeide på.	☐	☐	☐	☐	☐
27. Vi har evne til å stille spørsmål ved ”vedtatte” sannheter.	☐	☐	☐	☐	☐
28. Vi er svært fleksible i forhold til hvordan arbeidsoppgavene utføres.	☐	☐	☐	☐	☐
29. Vi spesifiserer ikke arbeidsoppgaver mer detaljert enn det som er absolutt nødvendig.	☐	☐	☐	☐	☐
30. Vi utnytter aktivt variasjonen i de ansattes kompetanse og interesser til å håndtere variasjonen og kompleksiteten til bedriftens omgivelser.	☐	☐	☐	☐	☐

Medvirkning fra utviklerne:	**Sterkt uenig**	**Uenig**	**Hverken enig eller uenig**	**Enig**	**Sterkt Enig**
31. Utviklerne har stor innflytelse på organiseringen av det arbeidet de selv utfører.	☐	☐	☐	☐	☐
32. Utviklerne deltar aktivt med forslag til forbedring av utviklingsarbeidet.	☐	☐	☐	☐	☐
33. Utviklerne deltar aktivt i utforming av rutiner og prosedyrer for utvikling.	☐	☐	☐	☐	☐
34. Prinsipper for utvikling av programvare blir jevnlig diskutert blant utviklerne.	☐	☐	☐	☐	☐
35. Utviklerne har ansvarsoppgaver knyttet til bedriftens forbedringsarbeid.	☐	☐	☐	☐	☐
36. Utviklerne deltar aktivt i å sette konkrete mål for forbedringsarbeidet.	☐	☐	☐	☐	☐
37. Forbedring av utviklingsprosesser blir jevnlig diskutert blant utviklerne.	☐	☐	☐	☐	☐
38. Utviklerne deltar aktivt i arbeidet med å planlegge nye prosjekter.	☐	☐	☐	☐	☐

Vennligst marker hvor godt du mener følgende utsagn beskriver oppnådde resultater av forbedrings-arbeid i *din* organisasjon.

Resultater:	**Sterkt uenig**	**Uenig**	**Hverken enig eller uenig**	**Enig**	**Sterkt Enig**
1. Forbedringsarbeidet vårt har gitt en betydelig økning i generell utviklingskompetanse.	☐	☐	☐	☐	☐
2. Forbedringsarbeidet vårt har ført til at vi gjør en vesentlig bedre jobb.	☐	☐	☐	☐	☐
3. I løpet av de siste tre årene har vi oppnådd vesentlige kostnadsreduksjoner i utvikling av programvare.	☐	☐	☐	☐	☐
4. I løpet av de siste tre årene har vi oppnådd vesentlige reduksjoner i utviklingstid for programvare.	☐	☐	☐	☐	☐
5. I løpet av de siste tre årene har vi oppnådd vesentlig økning i kundetilfredshet.	☐	☐	☐	☐	☐

Generell informasjon

Hvor mange ansatte er det i din bedrift? _________

Hvor mange av disse jobber med programvareutvikling? _________

Hva er det primære markedssegment til bedriften? (marker ett valg)

☐ Bank/finans/forsikring ☐ Offentlig sektor ☐ Industri ☐ Elektronikk/telekom

☐ Annet: ____

Hvilken rolle har programvareprosjekter i din bedrift? (marker ett valg)

☐ Utvikling/salg av hyllevare ☐ Utvikling for eksterne kunder ☐ Utvikling for internt bruk

På en skala fra 1 til 7, hvordan vil du karakterisere bedriftens omgivelser?

Stabil <---> Ustabil

1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐

Uforutsigbar <---> Forutsigbar

1	2	3	4	5	6	7
☐	☐	☐	☐	☐	☐	☐

Har bedriften et aktivt forbedrings- eller kvalitetsprogram? ☐ **Ja** ☐ **Nei**

Hvis JA, hvor mange år er det siden oppstart av dette programmet? ___________

Modeller og standarder som er i aktiv bruk i bedriften? (marker alle som er i bruk)

☐ ISO 15504/SPICE ☐ SEI CMM/IDEAL ☐ Bootstrap

☐ ISO 9001 (9000-3) ☐ EFQM (TQM) ☐ ESSI/SPIQ

☐ Annet: ______________________________

Hva er din jobbtittel? ______________________________

Hva er din høyeste fullførte utdanning?

☐ Ing. høyskole/cand. mag. eller tilsvarende ☐ Sivilingeniør/hovedfag eller tilsvarende

☐ Doktorgrad ☐ Annet: __________________________

Hvor mange års erfaring har du med programvareutvikling? ______________

Hvor mange år har du arbeidet i denne bedriften? ______________

Tusen takk for at du tok deg tid til å svare på disse spørsmålene!

Appendix G: Survey Data

Id	Exploration						Leadership					Business					Measurement						Exploration								Participation						
M02	1	2	3	2	2	3	4	4	3	4	3	4	3	3	4	4	4	4	3	3	4	4	4	5	3	3	4	4	4	3	4	3	4	3	2	2	3
M03	2	2	4	2	3	4	3	4	4	4	4	2	2	2	2	2	4	4	2	3	3	4	2	3	2	2	4	2	2	4	4	4	4	3	2	2	3
M05	3	3	4	2	4	2	4	3	4	4	3	2	3	4	4	3	2	2	2	2	2	4	4	5	5	5	4	4	4	4	5	5	4	3	3	3	3
M06	3	4	4	3	3	4	3	3	4	3	3	3	2	3	3	2	3	3	3	3	3	3	3	4	4	3	3	2	3	3	4	3	4	4	3	2	3
M07	4	4	5	4	3	1	4	4	5	5	4	4	4	4	4	3	5	4	4	4	4	4	3	4	4	3	4	4	4	3	3	4	4	4	4	3	4
M08	4	4	4	3	3	3	5	4	5	4	5	4	4	4	4	4	4	4	4	4	3	3	4	5	5	4	4	3	3	4	4	4	4	4	4	3	3
M08	4	4	4	4	5	2	4	4	4	4	4	4	2	4	4	2	4	3	2	2	2	5	4	4	4	4	4	3	3	3	4	4	4	4	4	4	4
M09	4	3	4	3	4	3	4	4	5	4	4	5	5	4	5	4	5	5	4	4	4	4	4	4	5	4	4	5	4	5	4	4	4	4	4	4	3
M10	2	4	4	1	3	3	3	3	5	1	3	3	3	2	3	3	5	2	1	1	1	3	3	5	2	4	3	4	3	4	5	4	4	4	3	3	4
M11	4	4	4	4	4	2	4	4	4	4	4	4	3	4	4	3	4	4	4	4	4	4	3	4	3	3	4	3	3	3	3	4	4	4	4	4	4
M12	4	3	4	2	4	2	1	2	2	2	2	2	4	3	2	3	2	2	2	2	4	4	3	4	4	3	3	4	4	3	4	4	4	4	4	3	3
M13	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	4	3	3	4	3	3	3	3	3	3	3	4	4	3	4	4	3
M16	2	2	2	2	3	3	4	4	4	4	3	3	3	4	3	3	4	2	1	2	2	4	4	4	3	3	4	4	4	2	4	4	4	4	4	2	4
M17	4	4	4	3	4	3	5	5	5	5	4	4	3	4	5	3	4	3	2	2	2	3	4	4	4	4	4	4	4	3	5	5	5	5	4	4	4
M18	4	4	4	4	3	3	4	4	5	4	4	4	4	4	4	4	5	4	3	3	3	3	3	4	3	3	3	4	4	3	4	4	3	4	3	3	3
M20	4	4	5	4	4	4	4	4	4	4	5	3	4	4	4	3	4	4	3	3	3	4	3	4	4	2	3	2	2	3	4	4	3	4	3	3	4
M21	4	4	5	4	4	3	4	4	4	4	3	3	3	3	4	4	4	4	3	4	4	4	4	3	3	3	4	4	4	4	4	3	4	3	3	3	4
M21	4	4	4	4	3	3	4	4	4	4	4	5	4	4	5	3	3	3	2	2	3	3	5	3	3	3	3	2	3	2	3	3	4	3	4	3	3
M22	4	3	5	4	3	4	4	4	4	3	3	4	2	4	4	3	3	4	3	3	4	4	2	2	2	3	3	2	3	2	2	3	4	4	4	2	4
M22	3	4	4	5	4	3	4	4	4	4	3	4	3	4	3	3	5	5	5	5	5	4	3	3	3	2	4	2	3	2	3	2	3	3	4	3	3
M23	3	2	3	3	3	2	3	4	4	3	3	2	2	3	3	3	4	3	3	3	4	3	3	4	3	3	3	3	3	3	4	4	3	4	3	2	3
M24	2	4	4	3	4	3	4	4	4	2	2	2	2	3	3	3	4	2	2	2	2	4	4	4	4	2	4	3	4	2	2	4	4	4	2	2	4
M24	3	4	4	2	3	3	3	2	3	2	4	2	3	4	4	2	3	2	2	2	2	3	3	4	3	3	3	4	3	3	4	3	5	3	2	2	3
M24	2	3	4	2	2	2	5	2	3	2	3	2	1	3	1	2	4	2	2	3	2	3	3	4	4	4	4	4	5	3	5	3	4	3	2	2	3
M25	4	3	4	4	3	4	3	3	4	3	2	2	2	3	3	3	4	4	3	4	4	4	4	4	4	5	4	5	4	4	4	4	4	3	4	2	2
M26	3	3	5	3	4	4	4	4	5	4	4	2	3	4	4	3	5	4	2	3	4	3	3	4	3	3	3	4	4	4	4	4	4	4	3	4	5
M27	4	5	4	3	4	2	4	4	4	4	4	3	4	3	4	3	4	3	3	3	3	3	4	4	5	4	4	5	4	4	5	5	4	4	5	4	4
M28	1	2	4	4	4	2	3	4	4	2	4	4	2	4	3	3	4	4	2	3	3	4	2	4	4	4	4	2	2	2	4	2	4	5	4	2	4
M30	2	4	4	4	3	2	4	4	4	4	3	4	3	3	5	3	5	5	4	5	4	5	3	5	5	4	4	3	4	4	4	4	4	4	3	4	4
M31	3	4	4	4	4	2	4	3	3	3	3	2	3	4	4	3	4	4	3	3	3	3	2	3	2	2	2	2	2	2	4	4	4	3	3	3	3
M32	3	2	2	3	3	4	3	3	4	3	2	4	2	2	3	3	4	4	2	2	3	2	3	3	2	2	3	4	4	3	4	3	2	3	3	2	2
M33	3	3	4	3	3	2	4	4	3	3	3	4	3	4	4	3	5	4	3	3	4	4	2	3	3	3	4	3	2	2	3	2	2	4	3	3	4
M34	4	4	5	5	4	2	4	4	5	5	5	5	5	4	5	4	5	4	5	5	5	5	5	5	5	5	5	4	4	5	4	4	5	5	5	4	4
M35	2	4	4	3	4	2	3	4	4	3	4	4	4	4	5	3	4	2	2	2	2	4	4	4	4	4	3	3	4	3	4	4	4	4	3	3	4
M36	2	4	4	3	2	2	3	3	4	3	5	2	4	4	3	3	4	4	2	2	3	4	4	4	4	3	4	4	4	3	4	4	4	4	3	2	4

Id	Exploration						Leadership					Business					Measurement						Exploration								Participation						
M37	3	2	3	2	2	2	2	2	2	2	2	2	3	2	2	2	3	3	2	2	2	4	3	3	2	2	3	4	4	2	4	3	2	4	4	4	2
M38	2	4	4	4	4	4	4	3	3	3	2	4	2	2	4	4	4	4	4	4	3	4	2	2	2	2	4	3	2	4	4	3	4	3	4	3	2
M39	4	4	4	4	4	1	2	2	2	3	2	4	4	4	4	3	4	4	2	2	2	4	4	5	5	4	4	4	4	4	4	3	4	4	4	3	3
M40	4	4	4	3	4	2	4	4	4	3	4	3	3	4	4	3	4	4	2	2	2	4	2	3	3	4	4	2	3	4	4	4	4	3	4	3	4
M40	3	2	4	2	3	2	4	4	4	4	3	4	4	4	4	3	4	2	2	2	4	4	4	4	3	3	3	4	4	4	4	4	4	3	4	3	3
M40	4	5	5	4	5	2	5	5	5	5	5	5	5	5	4	5	4	4	4	4	4	4	4	4	4	3	4	3	4	4	5	5	5	4	5	4	5
M41	4	4	5	4	2	3	4	4	5	4	5	4	3	4	3	3	4	5	2	3	3	4	3	2	3	2	2	1	3	3	4	3	3	2	3	2	3
M41	2	4	4	3	3	4	4	4	4	3	4	4	2	3	3	3	5	4	2	2	3	4	4	4	2	2	3	3	4	2	4	2	3	4	4	2	3
M41	2	3	4	3	2	2	4	4	4	4	4	4	2	4	3	4	4	4	4	4	3	3	3	4	4	4	5	4	3	4	3	4	4	2	4	4	3
M41	2	3	4	3	4	2	5	5	5	3	5	5	3	3	4	4	5	5	3	3	3	4	3	3	4	2	4	4	4	3	5	5	3	4	3	2	4
M41	3	2	4	3	2	2	4	3	4	4	3	4	3	2	3	2	4	4	2	2	2	3	2	2	4	4	2	2	3	2	2	2	2	2	3	2	3
M41	2	2	4	2	3	2	4	4	4	4	4	4	2	3	2	3	4	4	2	2	2	2	3	3	3	2	2	3	2	3	4	2	4	2	4	2	2
M41	3	3	4	2	3	4	4	3	4	4	3	4	3	4	2	3	4	4	3	4	3	4	3	4	4	3	3	4	2	3	4	3	2	4	3	2	2
M41	2	4	4	2	2	4	4	4	4	4	4	2	2	4	3	3	4	3	3	2	3	4	2	4	2	2	4	3	2	2	4	4	4	3	4	3	3
M41	4	4	4	5	3	2	5	4	5	3	2	3	3	4	3	3	4	4	3	3	5	4	4	5	5	5	4	4	4	4	5	5	5	4	4	4	4
M41	2	3	3	4	4	3	3	4	4	3	4	5	4	5	4	2	5	4	2	4	3	4	4	4	4	3	4	3	3	3	3	4	4	3	4	3	3
M41	3	4	4	4	3	2	4	4	4	3	4	4	4	4	4	3	4	5	2	3	3	3	3	3	4	3	3	3	2	4	3	3	4	3	3	3	3
M41	2	3	4	3	4	5	4	4	5	4	5	4	3	4	4	4	4	4	3	3	4	2	4	2	2	1	3	2	4	3	4	4	4	3	3	4	2
M41	4	4	5	3	4	2	5	5	5	5	5	5	5	4	5	4	3	3	3	3	3	4	3	4	4	4	4	3	3	3	5	4	5	5	5	4	4
M41	4	4	4	4	4	3	4	4	4	4	4	4	4	4	3	3	4	4	3	4	3	4	3	4	3	3	4	3	3	4	4	4	3	4	4	4	4
M42	4	3	4	3	4	2	3	2	3	3	2	2	3	4	4	2	2	3	3	3	4	3	3	4	4	4	4	3	3	4	4	4	4	4	3	4	4
M44	4	4	4	2	3	2	4	4	4	3	4	4	4	4	3	4	4	4	3	3	3	4	4	3	4	3	3	4	4	4	3	4	4	4	4	4	3
M45	4	3	4	3	3	3	3	3	3	3	2	2	2	2	2	2	3	2	2	2	2	3	4	4	4	4	4	3	3	3	4	4	3	4	2	2	3
M45	4	4	4	2	4	4	4	3	4	4	2	2	2	3	3	3	4	4	4	3	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	3	4
M46	4	4	4	3	4	3	4	4	4	4	4	3	3	4	3	3	4	3	3	3	4	4	3	4	3	3	4	4	4	4	4	4	4	4	4	4	4
M47	3	4	4	3	3	3	4	5	5	4	5	4	5	5	5	4	5	5	3	4	4	4	3	3	3	2	3	3	3	3	3	3	3	4	4	3	4
M49	4	4	3	5	4	2	4	2	3	2	3	4	3	4	4	2	4	4	2	1	4	4	2	3	4	4	4	3	3	3	4	4	4	4	5	4	5
M49	2	3	4	2	2	4	3	3	3	3	2	2	2	4	2	3	3	4	2	2	2	2	3	3	3	3	3	2	2	3	4	4	4	4	3	3	4
M50	3	4	4	3	3	2	3	2	3	3	2	4	4	4	4	4	4	4	4	4	4	4	3	4	3	4	3	3	3	4	3	3	3	3	3	3	3
M51	3	3	4	4	3	3	4	3	3	2	2	2	3	4	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	3	4	4	4	4	4	3	3
M53	5	4	3	4	1	3	3	3	5	2	3	4	4	3	3	2	3	5	4	4	4	2	4	4	4	4	3	5	3	4	3	4	5	4	3	3	4
M53	4	4	4	3	3	2	3	3	4	2	1	3	4	4	3	2	4	2	2	2	2	2	2	3	3	2	3	3	2	2	3	4	4	3	3	3	4
M53	2	3	4	4	3	5	2	2	3	3	3	3	2	3	3	2	4	3	1	1	2	2	3	3	3	3	4	3	4	3	3	3	3	3	2	3	2
M53	3	2	3	3	2	2	3	3	4	4	3	2	2	2	2	2	4	4	3	4	3	4	4	4	4	3	4	4	4	4	3	4	3	4	3	3	4
M54	2	2	3	1	1	2	4	4	3	2	3	3	2	2	3	2	4	4	3	4	3	2	3	4	3	3	3	4	3	4	4	4	4	4	4	3	3
M55	4	3	4	3	4	3	4	4	4	4	3	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	4	4	4	4	4	3	4	4	4	4
M56	4	4	4	3	3	2	4	4	4	3	3	2	3	4	3	2	4	2	2	2	3	3	3	4	4	3	4	4	4	4	4	4	4	4	4	4	4
M57	4	4	4	4	4	4	3	3	4	3	3	3	4	4	4	4	5	5	3	4	3	4	4	4	3	3	3	2	3	5	4	4	3	3	4	3	3
M58	4	4	4	5	4	2	4	5	5	5	5	4	4	4	4	3	4	5	4	4	5	5	3	5	4	4	4	3	3	3	4	5	4	5	5	4	4
M59	4	4	5	2	2	4	3	3	3	2	3	2	2	2	2	3	5	2	3	3	3	4	2	3	2	2	3	3	3	3	4	3	4	4	3	3	4
M60	4	3	5	3	3	2	3	3	3	4	3	2	2	4	3	4	2	4	2	3	3	4	4	5	5	4	3	4	5	4	5	5	5	3	3	4	3
M61	5	5	5	4	4	3	4	4	4	4	4	4	4	3	4	4	4	4	3	4	3	4	4	5	4	3	4	3	4	5	4	4	4	4	4	4	4
M61	3	4	4	4	3	4	4	4	4	4	4	3	4	3	4	3	4	4	3	3	4	4	3	4	3	3	4	3	3	3	4	4	4	4	3	4	3
M63	4	3	4	3	3	4	4	4	4	4	4	4	4	3	4	3	4	4	3	3	4	3	2	3	2	2	2	2	2	2	3	3	2	3	2	2	3
M65	3	4	4	4	3	2	4	3	4	3	4	3	3	3	3	3	3	3	3	4	3	2	4	4	4	4	4	4	4	4	4	4	4	5	3	3	4
M66	5	4	5	4	4	2	3	3	4	3	3	4	3	4	3	3	4	4	4	3	4	3	3	4	3	3	4	4	4	4	5	5	5	4	4	3	4
M67	2	4	4	3	2	2	2	2	4	2	2	4	3	4	4	3	4	4	3	3	4	3	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3
M69	4	4	4	3	3	3	4	4	4	3	3	2	3	4	3	4	3	3	3	3	4	3	4	4	4	3	4	3	4	4	3	4	3	3	3	2	3

Id	Exploration						Leadership					Business					Measurement						Exploration								Participation						
M69	2	3	3	3	2	2	2	2	4	3	2	2	2	4	2	2	4	5	4	4	2	3	2	4	4	3	3	4	4	2	4	4	2	3	2	2	3
M70	2	2	4	2	4	3	4	2	4	3	2	2	2	2	3	3	4	4	4	2	2	2	4	4	4	3	3	4	4	4	2	3	2	2	2	2	2
M71	2	3	4	3	2	3	3	4	4	4	4	4	3	3	2	3	5	4	3	4	4	4	3	4	4	3	3	3	3	4	4	4	4	3	3	4	4
M72	3	2	4	2	2	4	3	2	4	3	3	4	4	2	3	3	4	2	2	3	2	3	3	3	4	4	4	4	4	3	2	3	2	3	2	2	3
M73	2	2	4	4	2	2	2	2	4	2	3	2	2	2	2	3	4	2	2	2	2	2	2	3	3	3	3	3	2	2	2	4	4	4	2	2	4
M75	4	4	4	4	3	3	4	4	4	4	5	5	4	3	4	4	5	5	3	4	4	3	3	3	4	3	3	3	4	4	4	4	3	3	3	3	3
M76	2	3	4	3	3	2	4	4	4	3	3	4	2	2	4	4	4	3	4	4	4	2	2	4	4	4	4	4	4	2	4	3	2	4	2	4	3
M80	3	4	4	4	4	3	4	4	4	3	4	3	4	4	4	4	3	4	3	3	3	4	4	4	4	4	4	3	4	4	4	4	4	4	4	3	4
M81	2	4	4	3	2	3	2	2	4	3	4	4	4	3	4	4	4	2	2	2	3	3	2	4	4	3	4	4	4	2	4	4	4	4	3	2	3
M82	2	2	3	2	3	2	2	2	3	4	4	4	3	4	4	3	3	2	2	2	2	2	3	4	4	4	3	4	3	3	4	3	4	4	3	4	4
M83	2	3	4	3	4	3	4	4	4	3	4	4	2	4	4	4	4	4	3	3	4	4	3	2	3	3	3	2	3	3	4	4	4	3	4	3	4
M84	4	4	5	4	3	3	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	4	3	4	3	2	3	3	3	3	3	4	4	3	2	3	2
M84	3	4	5	3	4	3	4	4	4	3	3	5	4	4	5	4	5	5	5	5	4	3	3	4	3	3	4	3	3	4	4	4	3	3	4	5	3
M84	3	4	4	4	3	3	4	3	4	3	3	4	4	4	4	3	4	5	2	3	2	2	3	4	4	3	4	4	4	3	5	5	5	4	4	4	4
M86	4	4	5	4	3	3	4	4	4	4	3	3	3	3	4	3	4	4	3	3	4	4	3	4	3	3	3	3	3	3	4	3	3	4	3	3	3
M87	4	3	4	3	4	2	4	4	5	4	5	4	4	3	4	3	4	4	3	3	3	3	4	4	4	4	4	4	4	4	4	3	3	3	4	3	4
M102	3	4	4	4	4	3	4	4	4	4	4	4	3	3	4	3	4	4	3	3	3	3	4	4	3	3	4	4	4	4	4	3	4	3	4	3	3
M103	2	1	3	2	2	3	2	4	4	2	4	2	2	4	4	2	4	2	2	2	4	3	5	4	4	4	4	4	4	4	4	4	4	5	4	4	5
M105	2	3	3	3	3	2	4	4	4	4	4	4	5	5	4	4	4	4	3	3	4	3	4	4	4	4	4	4	4	5	4	4	4	4	5	4	4
M107	4	2	3	2	3	4	4	3	5	4	3	4	3	3	4	3	4	3	2	3	3	4	3	4	3	3	4	4	5	4	4	4	4	3	4	3	4
M112	3	2	3	2	2	3	4	3	4	3	3	3	3	2	3	3	3	2	2	2	2	3	3	3	3	3	3	3	3	3	3	2	3	2	2	2	2
M117	3	3	4	3	3	3	3	2	3	3	3	4	3	3	4	3	3	2	2	2	3	3	2	3	3	2	2	3	3	3	4	3	3	3	4	3	3
M119	4	3	4	3	3	2	3	3	3	3	3	2	2	3	3	2	2	2	2	2	2	3	3	4	4	4	3	4	4	4	5	4	4	4	3	3	3
M120	5	5	4	4	5	4	4	4	4	4	3	2	3	5	5	3	3	3	3	3	4	4	5	5	4	4	5	5	4	4	5	4	5	4	5	4	4
M122	3	4	4	3	3	3	4	4	4	4	4	3	3	3	4	3	4	2	3	3	3	3	3	4	3	3	4	4	4	4	5	5	4	3	3	3	3
P01	4	4	4	4	3	5	4	3	2	3	3	3	2	4	5	4	4	3	3	2	2	4	4	5	5	4	4	5	5	4	4	4	3	3	2	2	3
P02	4	3	4	2	4	4	5	5	5	5	4	4	2	3	5	3	3	1	1	1	2	2	4	3	4	4	2	3	5	3	4	3	4	4	2	2	2
P03	3	3	3	4	3	4	4	4	4	3	4	4	3	4	4	4	4	3	3	3	3	2	3	4	4	4	4	3	3	4	2	3	3	4	4	3	2
P04	2	2	4	3	1	3	3	2	2	1	3	3	2	2	3	2	4	4	2	3	2	4	2	2	3	1	2	4	4	1	3	4	3	4	2	1	3
P05	3	2	4	3	3	3	2	2	3	2	3	3	2	2	2	2	4	3	2	2	2	3	3	4	4	3	4	4	4	3	4	4	3	4	3	2	4
P06	2	3	4	4	3	4	3	4	4	3	3	4	4	3	4	2	5	4	4	4	4	4	4	5	4	2	5	3	4	4	2	3	3	3	4	4	2
P07	2	2	4	2	2	4	2	3	3	2	3	2	2	2	3	3	2	4	2	2	4	2	2	4	4	4	4	4	3	2	3	4	3	4	4	4	3
P08	3	3	4	3	2	3	4	4	4	4	3	3	3	4	4	3	4	4	3	3	3	2	4	4	4	3	3	3	4	4	4	3	3	4	3	3	4
P09	2	4	4	4	3	4	4	4	4	4	4	3	3	3	4	3	4	4	3	3	3	3	4	4	4	4	3	4	3	3	3	3	3	4	3	3	3
P10	1	2	3	2	4	3	3	3	3	2	3	1	2	3	2	2	2	3	3	3	2	4	3	3	4	4	3	3	2	3	3	4	2	4	2	2	3
P11	3	3	4	1	2	2	2	2	3	1	2	3	3	3	3	2	4	4	2	2	2	2	3	3	2	2	2	3	4	2	2	3	3	2	2	2	2
P12	1	1	2	1	2	3	2	1	2	1	1	1	1	2	1	1	3	3	2	2	1	4	3	4	4	3	4	4	5	4	4	3	2	4	1	1	4

Id	Performance					Size	Industry	Product business	Environm.		QA	Job function	Edu.	Sw exp.	Org. exp.
M02	3	4	2	3	4	35	3	2	3	3	0	2	3	4	5
M03	3	3	3	2	2	10	4	1	5	5	1	2	3	15	2
M05	4	4	3	3	4	27	4	2	3	3	0	2	3	15	5
M06	3	3	3	3	3	5	4	1	2	2	1	2	1	18	15
M07	4	4	4	4	4	8	3	2	2	2	1	1	2	0	8
M08	4	5	4	3	4	200	3	2	3	3	1	2	2	10	8
M08	5	4	5	4	4	200	3	2	2	5	1	2	2	7	11
M09	4	4	4	4	3	25	4	2	6	5	1	1	1	??	21
M10	3	4	3	3	3	7	0	2	1	3	1	2	2	12	6
M11	4	4	4	4	4	30	3	2	1	1	1	2	2	21	1
M12	3	3	4	4	3	26	4	2	4	4	1	2	2	11	11
M13	4	4	3	3	3	30	4	2	2	2	1	2	2	11	2
M16	4	3	4	3	4	30	4	2	2	3	1	2	3	10	20
M17	5	5	3	3	3	12	4	2	6	6	1	2	2	11	11
M18	4	4	3	3	4	26	3	1	4	3	1	2	2	2	3
M20	4	4	2	4	4	50	4	2	6	6	0	2	2	7	6
M21	4	4	4	2	4	60	4	2	4	4	1	2	2	18	10
M21	4	4	3	4	5	55	4	2	6	5	1	1	2	6	9
M22	4	5	5	4	5	250	4	2	3	4	1	2	2	15	15
M22	4	4	4	4	4	250	4	2	5	6	1	2	2	11	11
M23	4	4	2	2	3	5	0	2	5	5	0	2	2	4	7
M24	3	2	2	2	4	20	4	2	2	5	1	2	1	8	8
M24	3	3	2	2	3	25	4	2	2	5	1	2	2	8	29
M24	3	3	2	2	3	25	4	1	6	6	0	1	1	0	5
M25	4	4	4	3	3	4	4	1	5	6	1	2	2	8	8
M26	4	5	3	3	4	40	4	1	6	5	1	2	2	14	14
M27	4	4	4	3	4	11	4	2	5	6	0	2	2	9	9
M28	4	3	2	2	2	450	4	1	6	5	1	1	2	7	10
M30	5	5	5	5	4	800	4	2	6	4	1	1	1	3	7
M31	4	4	3	3	4	300	4	2	6	6	1	1	2	7	4
M32	3	3	3	4	4	660	4	1	3	3	1	1	1	10	20
M33	4	4	3	4	3	50	4	2	5	3	1	1	1	6	6
M34	5	5	3	3	5	30	4	1	5	5	1	1	2	0	1
M35	4	4	3	3	4	30	4	2	3	4	1	2	1	10	3
M36	3	3	3	3	4	5	4	3	6	6	0	2	2	10	4
M37	4	4	2	2	4	6	0	1	5	6	1	2	1	16	16
M38	4	4	3	4	3	200	4	1	2	2	1	2	2	22	10
M39	4	4	3	3	4	600	4	2	3	3	1	2	2	10	10
M40	4	4	3	3	3	450	4	1	3	4	1	2	2	15	15
M40	3	3	3	2	3	450	4	2	4	4	1	2	1	9	8
M40	4	5	3	4	4	600	4	1	5	4	1	2	2	12	12
M41	4	4	3	3	3	600	4	2	2	3	1	2	2	3	3
M41	3	3	3	4	4	600	4	2	5	5	1	2	1	8	2
M41	3	4	3	3	4	500	4	2	6	6	1	2	2	3	9
M41	4	3	3	3	3	600	4	2	6	2	1	2	1	0	1
M41	3	4	3	2	4	600	4	2	3	4	1	2	2	15	15

Id	Performance					Size	Industry	Product business	Environm.		QA	Job function	Edu.	Sw exp.	Org. exp.
M41	4	4	2	2	3	600	4	1	3	5	1	2	2	8	6
M41	4	3	3	3	2	600	4	2	4	5	1	2	2	7	10
M41	4	3	3	3	3	500	4	1	6	5	1	2	2	0	1
M41	4	4	5	3	4	26	4	1	4	5	1	2	1	12	28
M41	4	4	3	4	4	700	4	2	6	6	1	1	1	10	14
M41	4	4	4	4	3	500	4	1	4	6	1	2	1	4	16
M41	4	4	4	4	3	600	4	2	6	6	1	2	1	1	10
M41	4	5	4	5	4	500	3	2	6	6	1	2	1	13	12
M41	4	4	4	4	4	500	4	2	5	5	1	2	2	15	3
M42	4	4	3	3	4	50	3	2	2	3	0	2	1	7	7
M44	4	4	3	3	3	100	3	1	4	3	1	2	2	14	3
M45	3	3	3	3	3	30	4	2	3	3	0	2	2	17	10
M45	4	4	4	4	4	50	4	2	5	5	1	2	2	17	6
M46	4	4	3	3	4	18	4	2	2	3	1	2	2	9	4
M47	5	4	4	4	3	700	4	2	6	5	1	1	2	4	2
M49	4	4	3	3	4	30	4	2	6	5	0	2	2	5	5
M49	2	2	4	3	3	40	3	2	5	3	0	2	1	8	11
M50	4	3	4	3	5	500	4	2	2	2	1	1	1	20	4
M51	4	4	4	4	3	300	3	2	2	3	1	1	2	0	32
M53	2	3	3	3	5	50	0	2	5	3	0	2	1	15	2
M53	3	3	2	3	3	50	0	2	4	5	0	2	2	2	2
M53	3	3	2	2	2	65	1	2	6	6	1	2	2	15	3
M53	3	3	4	3	3	65	0	2	5	5	0	2	2	7	7
M54	3	3	3	3	3	50	3	2	2	3	0	1	2	25	7
M55	4	4	4	4	4	16	1	2	4	5	1	2	2	36	16
M56	4	4	3	2	3	18	2	2	5	5	1	2	2	17	1
M57	4	4	4	4	3	100	1	1	2	2	1	2	2	18	5
M58	4	4	4	4	4	35	0	1	1	2	1	2	2	20	6
M59	3	3	3	3	4	30	1	2	3	3	0	2	1	29	4
M60	2	3	4	4	3	36	2	1	2	2	0	2	1	13	2
M61	4	4	4	4	4	350	3	2	2	3	0	2	3	8	3
M61	4	4	4	4	4	380	3	2	4	4	1	2	2	25	6
M63	4	4	2	4	4	140	1	2	3	2	1	2	1	12	2
M65	3	4	3	4	4	110	1	2	3	3	1	2	1	15	3
M66	4	4	3	4	4	60	0	2	5	6	0	2	2	12	8
M67	3	4	2	2	3	400	4	2	6	2	1	1	2	5	2
M69	4	4	4	3	3	250	1	2	4	3	0	2	2	16	9
M69	2	2	1	3	3	120	1	2	5	5	1	1	2	0	1
M70	3	3	3	3	4	120	0	2	2	3	0	2	1	6	6
M71	3	4	4	3	3	100	1	3	2	2	0	2	2	5	10
M72	2	3	3	3	3	600	4	3	2	4	1	2	1	7	7
M73	3	4	3	3	4	250	4	2	6	3	1	2	2	11	3
M75	4	4	4	4	3	200	1	2	5	5	1	1	1	11	14
M76	3	3	4	3	4	480	4	2	5	4	1	1	0	19	4
M80	4	4	3	5	4	7	1	1	5	5	0	2	1	23	1
M81	3	3	4	4	5	60	4	3	6	6	1	2	1	15	13
M82	3	4	3	3	4	400	4	2	3	2	1	2	1	15	10

Id	Performance					Size	Industry	Product business	Environm.		QA	Job function	Edu.	Sw exp.	Org. exp.
M83	5	5	3	3	4	200	1	3	5	4	1	1	2	10	1
M84	4	4	3	3	3	50	2	2	2	2	1	2	0	30	30
M84	4	4	3	3	4	150	3	2	2	3	1	2	1	??	25
M84	4	4	4	4	3	500	4	2	3	3	1	2	1	25	18
M86	3	3	2	2	4	450	4	2	5	5	0	2	2	7	3
M87	4	4	4	4	4	550	4	2	5	3	1	2	1	15	2
M102	4	4	4	4	4	5	0	2	4	4	1	2	0	25	9
M103	4	4	5	5	4	55	3	1	5	2	0	2	2	2	2
M105	4	4	3	4	4	3	3	2	6	6	1	2	2	12	2
M107	3	4	3	3	4	4	3	1	5	6	0	2	2	12	8
M112	2	3	2	2	3	40	3	2	4	5	0	2	1	4	3
M117	3	3	3	3	3	3	0	2	2	3	1	2	2	5	5
M119	4	4	3	3	3	13	3	2	5	5	0	2	2	15	11
M120	4	4	3	4	4	1	3	2	6	6	0	2	1	20	15
M122	3	4	4	3	3	4	0	1	3	3	1	2	1	16	7
P01	3	3	3	3	4	110	2	2	2	4	1	2	2	4	4
P02	3	3	2	2	3	110	2	2	4	5	1	1	2	32	14
P03	4	4	3	4	3	600	4	1	3	3	1	1	2	6	17
P04	2	2	3	3	3	600	4	1	3	3	1	2	2	20	5
P05	2	3	2	3	4	14	2	2	3	3	0	2	1	20	10
P06	4	4	2	3	1	30	4	2	5	3	1	1	2	0	19
P07	4	3	1	1	2	30	4	2	6	6	1	2	2	4	4
P08	4	4	3	3	3	30	4	1	4	4	0	2	2	15	12
P09	3	4	4	3	3	30	4	1	4	3	1	1	2	6	3
P10	4	3	3	3	3	20	4	1	5	6	0	2	2	10	10
P11	2	3	3	3	3	30	4	2	3	3	0	2	2	13	7
P12	2	2	2	2	2	30	2	1	6	6	0	1	2	20	6

References

ABDEL-HAMID, T.K. (1996) The Slippery Path to Productivity Improvement, *IEEE Software*, Vol. 13, No.4, pp. 43-52.

ABDEL-HAMID, T.K. AND MADNICK, S.E. (1990) The Elusive Silver Lining: How We Fail to Learn from Software Development Failures, *Sloan Management Review*, Vol. 32, No. 1, Fall, pp. 39-48.

ACKERMAN, M.S. AND HALVERSON, C.A. (2000) Reexamining Organizational Memory, *Communications of the ACM*, Vol. 43, No. 1, pp. 58-64.

ADLER, P.S. AND BORYS, B. (1996) Two Types of Bureaucracy: Enabling and Coercive, *Administrative Science Quarterly*, Vol. 41, No. 1, pp. 61-89.

ADLER, P.S. AND WINOGRAD, T.A. (Eds.) (1992) *Usability - Turning Technologies into Tools*, New York: Oxford University Press.

AHIRE, S.L. AND GOLHAR, D.Y. (1996) Quality Management in Large vs. Small Firms: An Empirical Investigation, *Journal of Small Business Management*, Vol. 34, No. 2, pp. 1-13.

AHIRE, S.L., GOLHAR, D.Y., AND WALLER, M.A. (1996) Development and Validation of TQM Implementation Constructs, *Decision Sciences*, Vol. 27, No.1, pp. 23-56.

ANASTASI, A. AND URBINA, S. (1997) *Psychological Testing*, Seventh Edition, Upper Saddle River, New Jersey: Prentice-Hall.

ARDIS, M.A. AND MARCOLIN, B.L. (Eds.) (2001) *Diffusing Software Product and Process Innovations*, IFIP TC8 WG8.6 Fourth Working Conference on Diffusing Software Product and Process Innovations, April 7-10, Banff, Canada; Boston: Kluwer.

ARGYRIS, C. (1982) How Learning and Reasoning Processes Affect Organizational Change, in P.S. Goodman and Associates (Eds.), *Change in Organizations: New Perspectives on Theory, Research, and Practice*, San Francisco: Jossey-Bass.

ARGYRIS, C. (1999) *On Organizational Learning*, Second Edition, Oxford: Blackwell Business.

ARGYRIS, C. AND SCHÖN, D.A. (1978) *Organizational Learning: A Theory of Action Perspective*, Reading, Massachusetts: Addison-Wesley.

ARGYRIS, C. AND SCHÖN, D.A. (1996) *Organizational Learning II: Theory, Method, and Practice*, Reading, Massachusetts: Addison-Wesley.

ARISHOLM, E., SKANDSEN, J., SAGLI, K., AND SJØBERG, D. (1999) Improving an Evolutionary Development Process – A Case Study, *Proceedings of the European Software Process Improvement Conference (EuroSPI'99)*, Pori, Finland, 25-27 October.

ARMOUR, P.G. (2001) The Laws of Software Process, *Communications of the ACM*, Vol. 44, No. 1, pp. 15-17.

ASHBY, W.R. (1956) *An Introduction to Cybernetics*, London, UK: Chapman & Hall.

AUNE, A. (1985) *Kvalitetssirkler: Problemløsningsgrupper for personlig vekst, kvalitet og produktutvikling* (Quality Circles: Problem Solving Groups for Personal Growth, Quality, and Product Development), Oslo: Universitetsforlaget (in Norwegian).

AVISON, D., LAU, F., MYERS, M., AND NIELSEN, P.A. (1999) Action Research, *Communications of the ACM*, Vol. 42, No. 1, pp. 94-97.

BAETJER, H. JR. (1998) *Software as Capital: An Economic Perspective on Software Engineering*, Los Alamitos, California: IEEE Computer Society Press.

BAGOZZI, R.P. (1996) Measurement in Marketing Research: Basic Principles of Questionnaire Design, in R.P. Bagozzi (Ed.), *Principles of Marketing Research*, Cambridge, Massachusetts: Blackwell, pp. 1-49.

BALZER, R. (1981) Transformational Implementation: An Example, *IEEE Transactions on Software Engineering*, Vol. 7, No. 1, pp. 3-14.

BAROUDI, J. AND ORLIKOWSKI, W. (1989) The Problem of Statistical Power in MIS Research, *MIS Quarterly*, Vol. 13, No. 1, pp. 87-106.

BARRETT, F.J. (1998) Creativity and Improvisation in Jazz and Organizations: Implications for Organizational Learning, *Organization Science*, Vol. 9, No. 5, pp. 605-622.

BARUCH, Y. (1999) Response Rate in Academic Studies - A Comparative Analysis, *Human Relations*, Vol. 52, No. 4, pp. 421-438.

BASILI, V.R. (1989) Software Development: A Paradigm for the Future, *Proceedings of the 13th Annual International Computer Software & Applications Conference (COMPSAC)*, Keynote Address, Orlando, FL, September.

BASILI, V.R. (1993) The Experimental Paradigm in Software Engineering, in H.D. Rombach, V.R. Basili, and R. Selby (Eds), *Experimental Software Engineering Issues: Critical Assessment and Future Directives*, Proceedings of the Dagstuhl-Workshop, Germany, 1992, Lecture Notes in Computer Science, No. 706, Berlin: Springer-Verlag, pp. 3-12.

BASILI, V.R. (1996) The Role of Experimentation in Software Engineering: Past, Current, and Future, *Proceedings of the 18th International Conference on Software Engineering (ICSE-18)*, Berlin, Germany, March 25-29, pp. 442-449.

BASILI, V.R. AND CALDIERA, G. (1995) Improve Software Quality by Reusing Knowledge and Experience, *Sloan Management Review*, Vol. 37, No. 1, Autumn, pp. 55-64.

BASILI, V.R. AND ROMBACH, H.D. (1988) The TAME project: Towards improvement-oriented software environments, *IEEE Transactions on Software Engineering*, Vol. 14, No. 6, pp. 758-773.

BASILI, V.R. AND ROMBACH, H.D. (1991) Support for Comprehensive Reuse, *IEEE Software Engineering Journal*, Vol. 6, No. 5, pp. 303-316.

BASILI, V.R. AND SELBY, R.W. (1991) Paradigms for Experimentation and Empirical Studies in Software Engineering, *Reliability Engineering and System Safety*, Vol. 32, Nos. 1-2, pp.171-191.

BASILI, V.R. AND TURNER, A.J. (1975) Iterative Enhancement: A Practical Technique for Software Development, *IEEE Transactions on Software Engineering*, Vol. 1, No. 4, pp. 390-396.

BASILI, V.R. AND WEISS, D. (1984) A methodology for collecting valid software engineering data, *IEEE Transactions on Software Engineering*, Vol. 10, No. 6, pp. 728-738.

BASILI, V.R., CALDIERA, G., AND ROMBACH, H.D. (1994a) Experience Factory, in J.J. Marciniak (Ed.), *Encyclopedia of Software Engineering*, Vol. 1, John Wiley & Sons, pp. 469-476.

BASILI, V.R., CALDIERA, G., AND ROMBACH, H.D. (1994b) The Goal Question Metric Approach, in J.J. Marciniak (Ed.), *Encyclopedia of Software Engineering*, Vol. 1, John Wiley & Sons, pp. 528-532.

BASILI, V.R., CONDON, S.E., EL EMAM, K., HENDRICK, R.B., AND MELO, W. (1997) Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components, *Proceedings of the 19th International Conference on Software Engineering*, pp. 282-291.

BASILI, V.R., SELBY, R.W., AND HUTCHENS, D.H. (1986) Experimentation in Software Engineering, *IEEE Transactions on Software Engineering*, Vol. 12, No. 7, pp. 733-743.

BASKERVILLE, R. AND WOOD-HARPER, A.T. (1996) A Critical Perspective on Action Research as a Method for Information Systems Research, *Journal of Information Technology*, Vol. 11, No. 3, pp. 235-246.

BASKERVILLE, R. AND WOOD-HARPER, A.T. (1998) Diversity in Information Systems Action Research Methods, *European Journal of Information Systems*, Vol. 7, No. 2, pp. 90-107.

BASSMANN, M.J., MCGARRY, F., AND PAJERSKI, R. (1995) *The Software Measurement Guidebook*, Revision 1, Software Engineering Laboratory Series, SEL-94-002, Greenbelt, Maryland: Goddard Space Flight Center.

BATE, R. (Ed.) (1995) *A Systems Engineering Capability Maturity Model*, Version 1.1, Technical Report, CMU/SEI-95-MM-003, Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

BATESON, G. (1972) *Steps to an Ecology of Mind*, New York: Ballantine Books.

BATESON, G. (1979) *Mind and Nature: A Necessary Unity*, New York: Bantam Books.

BATISTA, J. AND DE FIGUEIREDO, A.D. (2000) SPI in a Very Small Team: a Case with CMM, *Software Process – Improvement and Practice*, Vol. 5, No. 4, pp. 243-250.

BAUM, L.F. (1995) *The Wizard of Oz*, London: Penguin Books (originally published as *The Wonderful Wizard of Oz*, George M. Hill Company, 1900).

BAUMGARTEL, H. (1959) Using Employee Questionnaire Results for Improving Organizations: The Survey 'Feedback' Experiment, *Kansas Business Review*, Vol. 12, pp. 2-6, December.

BECHTOLD, R.T. (1996) *Improving the Software Process through Process Definition and Modeling*, London: International Thomson Computer Press.

BECK, K. (1999) *Extreme Programming Explained: Embrace Change*, Reading, Massachusetts: Addison-Wesly.

BEER, M. EISENSTAT, R.A., AND SPECTOR, B. (1990) Why Change Programs Don't Produce Change, *Harvard Business Review*, Vol. 68, No. 6, pp. 158-166.

BENBASAT, I. AND ZMUD, R.W. (1999) Empirical Research in Information Systems: The Practice of Relevance, *MIS Quarterly*, Vol. 23, No. 1, pp. 3-16.

BENBASAT, I., GOLDSTEIN, D.K., AND MEAD, M. (1987) The Case Research Strategy in Studies of Information Systems, *MIS Quarterly*, Vol. 11, No. 3, pp. 369-386.

BENJAMIN, R. AND LEVINSON, E. (1993) A Framework for Managing IT-Enabled Change, *Sloan Management Review*, Vol. 34, No. 4, pp. 23-33.

BENNETTS, P.D.C., WOOD-HARPER, A.T., AND MILLS, S. (1999) The Soft System Methodology as a Framework for Software Process Improvement, in E. McGuire (Ed.), *Software Process Improvement: Concepts and Practices*, London: Idea Group Publishing, pp. 17-30.

BENNIS, W.G. AND NANUS, B. (1997) *Leaders: Strategies for Taking Charge*, Second Edition, New York: HarperBusiness.

BENSON, J.K. (1983) A Dialectic Method for the Study of Organizations, in G. Morgan (Ed.), *Beyond Method: Strategies for Social Research*, Newbury Park, California: Sage, pp. 331-346.

BENSON, P.G., SARAPH, J.V., AND SCHROEDER, R.G. (1991) The Effects of Organizational Context on Quality Management: An Empirical Investigation, *Management Science*, Vol. 37, No. 9, pp. 1107-1124.

BERGER P.L. AND LUCKMANN, T. (1966) *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*, Harmondsworth: Penguin Books.

BERLINER, P.F. (1994) *Thinking in Jazz: The Infinite Art of Improvisation*, Chicago, IL: University of Chicago Press.

BERREMAN, G. (1966) Anemic and Emetic Analyses in Social Anthropology, *American Anthropologist*, Vol. 68, pp. 346-354.

BERRY, M. AND JEFFERY, R. (2000) An Instrument for Assessing Software Measurement Programs, *Empirical Software Engineering*, Vol. 5, No. 3, November, pp.183-200.

BIJKER, W.E., HUGHES, T.P., AND PINCH, T.J. (1987) *The Social Construction of Technological Systems*, Cambridge, Massachusetts: The MIT Press.

BIRÓ, M. AND TULLY, C. (1999) The Software Process in the Context of Business Goals and Performance, in R. Messnarz, and C. Tully (Eds.) *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press, pp. 15-27.

BJERKNES, G. (1991) Dialectical Reflection in Information Systems Development, *Scandinavian Journal of Information Systems*, Vol. 3, pp. 55-77.

BJERKNES, G., DAHLBOM, B. *ET AL*. (1990) *Organizational Competence in System Development: A Scandinavian Contribution*, Lund, Sweden: Studentlitteratur.

BLACK, S.A. AND PORTER, L.J. (1996) Identification of the Critical Factors of TQM, *Decision Sciences*, Vol. 27, No. 1, pp.1-21.

BLALOCK, H.M. (1969) *Theory Construction: From Verbal to Mathematical Formulations*, Englewood Cliffs, New Jersey: Prentice-Hall.

BLAUNER, R. (1964) *Alienation and Freedom*, Chicago: The University of Chicago Press.

BOEHM, B.W. (1988) A Spiral Model of Software Development and Enhancement, *IEEE Computer*, Vol. 21, No. 5, pp. 61-72.

BOEHM, B.W. AND PAPACCIO, P.N. (1988) Understanding and Controlling Software Costs, *IEEE Transactions on Software Engineering*, Vol. 4, No. 10, pp. 1462-1477.

BOHM, D. AND PEAT, F.D. (2000) *Science, Order, and Creativity*, Second Edition, London: Routledge.

BOLLINGER, T. AND MCGOWAN, C. (1991) A Critical Look at Software Capability Evaluations, *IEEE Software*, Vol. 8, No. 4, pp. 25-41.

BOX, G.E.P., HUNTER, W.G., AND HUNTER J.S. (1978) *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*, New York: John Wiley & Sons.

BREDRUP, H. (1995) *Performance Measurement in a Changing Competitive Industrial Environment: Breaking the Financial Paradigm*, Doctoral Dissertation, Trondheim, Norway: Norwegian University of Science and Technology.

BRIAND, L.C., DIFFERDING, C.M., AND ROMBACH, H.D. (1996a) Practical Guidelines for Measurement-Based Process Improvement, *Software Process: Improvement and Practice*, Vol. 2, pp. 253-280.

BRIAND, L.C., EL EMAM, K., AND MELO, W.L. (1999) An Inductive Method for Software Process Improvement: Concrete Steps and Guidelines, in K. El Emam and N.H. Madhavji (Eds.) *Elements of Software Process Assessment and Improvement*, Los Alamitos, California: IEEE Computer Society Press, pp. 113-130.

BRIAND, L.C., EL EMAM, K., AND MORASCA, S. (1995) *Theoretical and Empirical Validation of Software Product Measures*, Technical Report, ISERN-95-03, International Software Engineering Research Network.

BRIAND, L.C., EL EMAM, K., AND MORASCA, S. (1996b) On the Application of Measurement Theory in Software Engineering, *Empirical Software Engineering*, Vol. 1, Issue 1, pp. 61-88.

BRODMAN, J.G. AND JOHNSON, D.L. (1994) What Small Businesses and Small Organizations Say About the CMM, *Proceedings of the Sixteenth International Conference on Software Engineering*, IEEE Computer Society Press, pp. 331-340.

BRODMAN, J.G. AND JOHNSON, D.L. (1995) Return on Investment (ROI) from Software Process Improvement as Measured by US Industry, *Software Process Improvement and Practice*, Pilot Issue, pp. 35-47.

BROOKS, F.P. (1975) *The Mythical Man-Month: Essays on Software Engineering*, Reading, Massachusetts: Addison-Wesley.

BROOKS, F.P. (1987) No Silver Bullet: Essence and Accidents of Software Engineering, *IEEE Computer*, Vol. 20, No. 4, pp.10-19.

BROOKS, F.P. (1995) *The Mythical Man-Month*, *Essays on Software Engineering*, Anniversary Edition, Reading, Massachusetts: Addison-Wesley.

BROWN, J.S. AND DUGUID, P. (1991) Organizational Learning and Communities of Practice: Toward a Unified View of Working, Learning, and Innovation, *Organization Science*, Vol. 2, No. 1, pp. 40-57.

BROWN, J.S. AND DUGUID, P. (2000a) Balancing Act: How to Capture Knowledge Without Killing It, *Harvard Business Review*, Vol. 78, No. 3, pp. 73-80.

BROWN, J.S. AND DUGUID, P. (2000b) *The Social Life of Information*, Boston, Massachusetts: Harvard Business School Press.

BRUNSSON, N. (1985) *The Irrational Organization: Irrationality as a Basis for Organizational Action and Change*, Chichester, U.K.: Wiley.

BRUNSSON, N. (1989) *The Organization of Hypocrisy: Talk, Decisions and Actions in Organizations*, New York: John Wiley & Sons.

BRYMAN, A. AND CRAMER, D. (1997) *Quantitative Data Analysis with SPSS for Windows: A Guide for Social Scientists*, London: Routledge.

BSI (2001) *The TickIT Guide*, Issue 5, British Standards Institution, BSI-DISC.

BUDLONG, F. AND PETERSON, J. (1995) *Software Metrics Capability Evaluation Guide*, The Software Technology Support Center, Ogden Air Logistics Center, Hill Air Force Base.

BURKE, W.W. (1994) *Organization Development: A Process of Learning and Changing*, Second Edition, Reading, Massachusetts: Addison-Wesley.

BURNSTEIN, I., HOMYEN, A., SUWANASSART, T., SAXENA, G., AND GROM, R. (1999) A Testing Maturity Model for Software Test Process Assessment and Improvement, *Software Quality Professional*, Vol. 1, No. 4, pp. 8-21.

BURRELL, G. AND MORGAN, G. (1979) *Sociological Paradigms and Organizational Analysis*, London: Heinemann Books.

BUZAN, T AND BUZAN, B. (2000) *The Mind Map Book*, Millenium Edition, London: BBC Books.

CAMP, R.C. (1993) Benchmarking: The Search for Industry Best Practices That Lead to Superior Performance, in W.F. Christopher and C.G. Thor (Eds.) *Handbook for Productivity, Measurement, and Improvement*, Portland: Productivity Press.

CAMPBELL, D.T. AND FISKE, D.W. (1959) Convergent and Discriminant Validation by the Multitrait-Multimethod Matrix, *Psychological Bulletin*, Vol. 56, pp. 81-105.

CAMPBELL, D.T. AND STANLEY, J.C. (1963) *Experimental and Quasi-Experimental Designs for Research*, Boston: Houghton Mifflin Company.

CARD, D. (1991) Understanding Process Improvement, *IEEE Software*, Vol. 8, No. 4, pp. 102-103.

CAREY, A. (1967) The Hawthorne Studies, *American Sociological Review*, Vol. 32, June, pp. 403-417.

CARMINES, E.G. AND ZELLER, R.A. (1979) *Reliability and Validity Assessment*, Sage University Paper series on Quantitative Applications in the Social Sciences, series no. 07-017, Newbury Park, California: Sage.

CARROLL, L. (1982) *Alice's Adventures in Wonderland; and, Through the Looking-Glass and What Alice Found There*, Oxford: Oxford University Press.

CASSIDY, A. AND GUGGENBERGER, K. (2000) *A Practical Guide to Information Systems Process Improvement*, CRC Press.

CATTELL, R.B. (1966) The Scree Test for the Number of Factors, *Multivariate Behavioral Research*, Vol. 1, pp. 245-276.

CHECKLAND P. AND SCHOLES, J. (1990) *Soft Systems Methodology in Practice*, Chichester: Wiley.

CHECKLAND P. AND SCHOLES, J. (1999) *Soft Systems Methodology in Action*, Chichester: Wiley.

CHECKLAND, P. (1981) *Systems Thinking, Systems Practice*, Chichester: Wiley.

CHECKLAND, P. (1999) *Systems Thinking, Systems Practice: Includes a 30-year Retrospective*, Chichester: Wiley.

CHOO, C.W. (1998) *The Knowing Organization: How Organizations Use Information to Construct Meaning, Create Knowledge, and Make Decisions*, New York: Oxford University Press.

CIBORRA, C.U. (1999) Theory of Information Systems Based on Improvisation, in W.L. Currie and R.D. Galliers (Eds.) (1999) *Rethinking Management Information Systems: An Interdisciplinary Perspective*, Oxford: Oxford University Press, pp. 136-155.

CLIFF, N.R. (1988) The Eigenvalues-greater-than-one Rule and the Reliability of Components, *Psychological Bulletin*, Vol. 103, pp. 276-279.

COALLIER, F., MCKENZIE, R.M., WILSON, J.F., AND HATZ, J. (1994) *Trillium: Model for Telecom Product Development and Support Process Capability*, Release 3.0, Bell Canada.

COHEN, H.B. (1998) The Performance Paradox, *Academy of Management Executive*, Vol. 12, No. 3, pp. 30-40.

COHEN, J. (1977) *Statistical Power Analysis for the Behavioral Sciences*, Revised Edition, New York: Academic Press.

COHEN, J. (1988) *Statistical Power Analysis for the Behavioral Sciences*, Second Edition, Hillsdale, New Jersey: Laurence Erlbaum.

COHEN, J. (1992) A Power Primer, *Psychological Bulletin*, Vol. 112, No.1, pp. 155-159.

COHEN, M.D. AND BACDAYAN, P. (1994) Organizational Routines are Stored as Procedural Memory: Evidence form a Laboratory Study, *Organization Science*, Vol. 5, No. 4, pp. 554-568.

COMREY A. (1973) *A First Course on Factor Analysis*, London: Academic Press.

CONRADI, R. AND DYBÅ, T. (2001) An Empirical Study on the Utility of Formal Routines to Transfer Knowledge and Experience, *Proceedings of the joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE)*, Vienna, Austria, 10-14 September.

CONRADI, R., FERNSTRÖM, C., AND FUGETTA, A. (1992) Towards a Reference Framework for Process Concepts, in J.-C. Derniame (Ed.), *Proceedings of the Second European Workshop on Software Process Technology (EWSPT'92)*, LNCS, Springer-Verlag, pp. 3-17.

CONSOLINI, L. AND FONADE, G. (1997) *The European Systems and Software Initiative – ESSI: A review of Current Results*, Final Version, The European Commission's Directorate General III, Industry.

CONSTANTINE, L.L. (1993) Work Organization: Paradigms for Project Management and Organization, *Communications of the ACM*, Vol. 36, No. 10, pp. 35-43.

COOK, S.D.N. AND BROWN, J.S. (1999) Bridging Epistemologies: The Generative Dance Between Organizational Knowledge and Organizational Knowing, *Organization Science*, Vol. 10, No. 4, pp. 381-400.

COOK, T.D. AND CAMPBELL, D.T. (1979) *Quasi-Experimentation: Design & Analysis Issues for Field Settings*, Boston: Houghton Mifflin Company.

COOPER, D.R. AND SCHINDLER, P.S. (1998) *Business Research Methods*, Sixth Edition, Boston: Irwin/McGraw-Hill.

COOPER, J., FISHER, M., AND SHERER, S.W. (1999) *Software Acquisition Capability Maturity Model (SA-CMM)*, Version 1.02, Technical Report, CMU/SEI-99-TR-002, Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

CORBETT, J.M. (1992) Work at the Interface: Advanced Manufacturing Technology and Job Design, in P.S. Adler and T.A. Winograd (Eds.), *Usability - turning technologies into tools*, Oxford University Press, pp. 133-163.

CRASK, M.R. AND FOX, R.J. (1987) An Exploration of the Interval Properties of 3 Commonly Used Marketing-Research Scales: A Magnitude Estimation Approach, *Journal of the Market Research Society*, Vol. 29, No. 3, pp. 317-339.

CRESWELL, J.W. (1994) *Research Design: Qualitative and Quantitative Approaches*, Thousand Oaks, California: Sage.

CRONBACH, L.J. (1951) Coefficient Alpha and the Internal Consistency of Tests, *Psychometrica*, Vol. 16, pp. 297-334, September.

CRONBACH, L.J. (1971) Test Validation, in R.L. Thorndike (Ed.), *Educational Measurement*, Second Edition, Washington: American Council on Education, pp. 443-507.

CROSBY, P.B. (1979) *Quality is Free: The Art of Making Quality Certain*, New York: McGraw-Hill.

CROSBY, P.B. (1984) *Quality Without Tears*, New York: McGraw-Hill.

CROSBY, P.B. (1996) *Quality Is Still Free: Making Quality Certain in Uncertain Times*, New York: McGraw-Hill.

CSIKSZENTMIHALYI, M. (1996) *Creativity: Flow and the Psychology of Discovery and Invention*, New York: HarperCollins.

CUMMINGS, T.G. AND WORLEY, C.G. (2001) *Organization Development and Change*, Seventh Edition, Cincinnati, Ohio: South-Western College Publishing.

CURTIS, B. HEFLEY, W.E., AND MILLER, S.A. (1995) *People Capability Maturity Model*, Version 1.0, Technical Report, CMU/SEI-95-MM-002, Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

CURTIS, B., KELLNER, M.I., AND OVER, J. (1992) Process Modeling, *Communications of the ACM*, Vol. 35, No. 9, pp. 75-90.

CURTIS, B., KRASNER, H., AND ISCOE, N. (1988) A Field Study of the Software Design Process for Large Systems, *Communications of the ACM*, Vol. 31, No. 11, pp. 1268-1287.

CUSUMANO, M.A. (1991) *Japan's Software Factories: A Challenge to U.S. Management*, New York: Oxford University Press.

CYERT, R.M. AND MARCH, J.G. (1963) *A Behavioral Theory of the Firm*, Englewood Cliffs, New Jersey: Prentice-Hall.

CYERT, R.M. AND MARCH, J.G. (1992) *A Behavioral Theory of the Firm*, Second Edition, Oxford, UK: Blackwell.

DAFT, R.L. AND WEICK, K.E. (1984) Toward a Model of Organizations as Interpretation Systems, *Academy of Management Review*, Vol. 9, No. 2, pp. 284-295.

DAHLBOM, B. AND MATHIASSEN, L. (1993) *Computers in Context: The Philosophy and Practice of Systems Design*, Oxford, UK: NCC Blackwell.

DAHLBOM, B. AND MATHIASSEN, L. (1997) The Future of Our Profession, *Communications of the ACM*, Vol. 40, No. 6, pp. 80-89.

DASKALANTONAKIS, M.K. (1992) A Practical View of Software Measurement and Implementation Experiences within Motorola, *IEEE Transactions on Software Engineering*, Vol. 18, No. 11, pp. 998-1010.

DAVENPORT, T.H. (1993) *Process Innovation: Reengineering Work through Information Technology*, Boston, Massachusetts: Harvard Business School Press.

DAVIS, A.M., BERSOFF, E.H., AND COMER, E.R. (1988) A Strategy for Comparing Alternative Software Development Life Cycle Models, *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, pp. 1453-1461.

DAVIS, D. (1996) *Business Research for Decision Making*, Fourth Edition, Belmont, California: Duxbury Press.

DAVIS, L. (1977) Enhancing the Quality of Work Life: Developments in the United States, *International Labour Review*, Vol. 116, July-August, pp. 53-65.

DEBOU, C. AND KUNTZMANN-COMBELLES, A. (2000) Linking Software Process Improvement to Business Strategies: Experience from Industry, *Software Process: Improvement and Practice*, Vol. 5, pp. 55-64.

DEBOU, C., COURTEL, D., LAMBERT, H.-B., FUCHS, N., AND HAUX, M. (1999) Alcatel's Experience with Process Improvement, in R. Messnarz and C. Tully (Eds.), *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press, pp. 281-301.

DEEPHOUSE, C., MUKHOPADHYAY, T., GOLDENSON, D.R., AND KELLNER, M.I. (1996) Software Processes and Project Performance, *Journal of Management Information Systems*, Vol. 12, No. 3, pp. 187-205.

DELANTY, G. (1997) *Social Science: Beyond Constructivism and Realism*, Buckingham: Open University Press.

DELONE, W.H. AND MCLEAN, E.R. (1992) Information Systems Success: The Quest for the Dependent Variable, *Information Systems Research*, Vol. 3, No. 1, pp. 60-95.

DEMARCO, T. (1982) *Controlling Software Projects: Management, Measurement and Estimation*, New York: Yourdon Press.

DEMARCO, T. AND LISTER, T. (1999) *Peopleware: Productive Projects and Teams*, Second Edition, New York: Dorset House Publishing.

DEMING, W.E. (1982) *Quality, Productivity, and Competitive Position*. Cambridge, Massachusetts: MIT Centre for Advanced Engineering Study, 1982.

DEMING, W.E. (1986) *Out of the Crisis*. Cambridge, Massachusetts: MIT Center for Advanced Engineering Study.

DENISON, D. AND SPREITZER, G. (1991) Organizational Culture and Organizational Development: A Competing Values Approach, in R. Woodman and W. Posmore (Eds.), *Research in Organizational Change and Development*, Vol. 5, Greenwich, Connecticut: JAI Press, pp. 1-22.

DENZIN, N.K. (1978) *Sociological Methods: A Sourcebook*, New York: McGraw-Hill.

DENZIN, N.K. AND LINCOLN, Y.S. (Eds.) (2000) *Handbook of Qualitative Research*, Second Edition, Thousand Oaks, California: Sage.

DERNIAME, J.-C., KABA, B.A., AND WASTELL, D. (Eds.) (1999) *Software Process: Principles, Methodology, and Technology*, Lecture Notes in Computer Science, Vol. 1500, Berlin: Springer-Verlag.

DEVELLIS, R.F. (1991) *Scale Development: Theory and Applications*, Newbury Park, California: Sage.

DEWEY, J. (1929) *The Quest for Certainty*, New York: Minton, Balch.

DEWEY, J. (1938) *Logic: The Theory of Inquiry*, New York: Holt and Company.

DIERKES, M., ANTAL, A.B., CHILD, J., AND NONAKA, I. (Eds.) (2001) *Handbook of Organizational Learning and Knowledge*, Oxford: Oxford University Press.

DION, R. (1993) Process Improvement and the Corporate Balance Sheet, *IEEE Software*, Vol. 10, No. 4, pp. 28-35.

DOHERTY, N.F. AND KING, M. (1998a) The Consideration of Organisational Issues during the Systems Development Process: An Empirical Analysis, *Behaviour and Information Technology*, Vol. 17, No. 1, pp. 41-51.

DOHERTY, N.F. AND KING, M. (1998b) The Importance of Organisational Issues in Systems Development, *Information Technology & People*, Vol. 11, No. 2, pp. 104-123.

DOWSON, M. (1993) Software Process Themes and Issues, *Proceedings of the Second International Conference on the Software Process*, IEEE Computer Society Press, pp. 54-62.

DREW, D. (1992) Tailoring the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) to a Software Sustaining Engineering Organization, *Proceedings of the International Conference on Software Maintenance*, pp. 137-144.

DUNAWAY, D.K. AND MASTERS, S. (1996) *CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description*, Technical Report, CMU/SEI-96-TR-007, Carnegie Mellon University, Software Engineering Institute.

DYBÅ, T. (1999) Assessment-based Software Process Improvement, *Telektronikk*, Vol. 95, No. 1, pp. 37-47.

DYBÅ, T. (2000a) An Instrument for Measuring the Key Factors of Success in Software Process Improvement, *Empirical Software Engineering*, Vol. 5, No. 4, December, pp. 357-390.

DYBÅ, T. (2000b) Improvisation in Small Software Organizations, *IEEE Software*, Vol. 17, No. 5, September-October, pp. 82-87.

DYBÅ, T. (2000c) Planning Organizational Improvement Programs, in T. Dybå (Ed.), *SPIQ – Software Process Improvement for better Quality: Methodology Handbook* (in Norwegian), IDI Report 2/2000, Trondheim, Norway: Norwegian University of Science and Technology, Faculty of Physics, Informatics, and Mathematics.

DYBÅ, T. (2000d) Planning Process Improvement Experiments, in T. Dybå (Ed.), *SPIQ – Software Process Improvement for better Quality: Methodology Handbook* (in

Norwegian), IDI Report 2/2000, Trondheim, Norway: Norwegian University of Science and Technology, Faculty of Physics, Informatics, and Mathematics.

DYBÅ, T. (Ed.) (2000e) *SPIQ – Software Process Improvement for better Quality: Methodology Handbook* (in Norwegian), IDI Report 2/2000, Trondheim, Norway: Norwegian University of Science and Technology, Faculty of Physics, Informatics, and Mathematics.

DYBÅ, T. AND MOE, N.B. (1999) Rethinking the Concept of Software Process Assessment, *Proceedings of the European Software Process Improvement Conference (EuroSPI'99)*, Pori, Finland, 25-27 October.

DYBÅ, T. AND SKOGSTAD, Ø. (1997) Measurement-Based Software Process Improvement, *Telektronikk*, Vol. 93, No. 1, pp. 73-82.

DYBÅ, T., STÅLHANE, T., AND PALMSTRØM R. (1997) Experience of Goal-Oriented Measurement using **ami** and GQM, *Proceedings of the 8th European Software Control and Metrics Conference (ESCOM'97)*, Berlin, Germany, 26-28 May.

EARL, M.J. (1996) Business Process Reengineering: A Phenomenon of Organization, in M.J. Earl (Ed.), *Information Management: The Organizational Dimension*, New York: Oxford University Press, pp. 53-76.

EDMONSON, A. AND MOINGEON, B. (1998) From Organizational Learning to the Learning Organization, *Management Learning*, Vol. 29, No. 1, pp. 5-20.

EFQM (1999) *The EFQM Excellence Model*, Brussels: European Foundation for Quality Management.

EHN, P. (1992) Scandinavian Design: On Participation and Skill, in P.S. Adler and T.A. Winograd (Eds.), *Usability – Turning Technologies into Tools*, New York: Oxford University Press, pp. 96-132.

EISENHARDT, K.M. (1989) Building Theories from Case Study Research, *Academy of Management Review*, Vol. 14, No. 4, pp. 532-550.

EISENHARDT, K.M. AND TABRIZI, B.N. (1995) Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry, *Administrative Sciences Quarterly*, Vol. 40, March, pp. 84-110.

EL EMAM, K. (1998) *The Internal Consistency of the ISO/IEC 15504 Software Process Capability Scale*, Technical Report, ISERN-98-06, International Software Engineering Research Network.

EL EMAM, K. AND BIRK, A. (2000) Validating the ISO/IEC 15504 Measures of Software Development Process Capability, *Journal of Systems and Software*, Vol. 51, No. 2, pp. 119-149, 2000.

EL EMAM, K. AND BRIAND, L. (1999) Costs and Benefits of Software Process Improvement, in R. Messnarz, and C. Tully (Eds.) *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press, pp. 177-197.

EL EMAM, K. AND GOLDENSON, D.R. (2000) An Empirical Review of Software Process Assessment, *Advances in Computers*, Vol. 53, pp. 319-423.

EL EMAM, K. AND MADHAVJI, N.H. (1995) The Reliability of Measuring Organizational Maturity, *Software Process – Improvement and Practice*, Vol. 1, No. 1, pp. 3-25.

EL EMAM, K. AND MADHAVJI, N.H. (1996) An Instrument for Measuring the Success of the Requirements Engineering Process in Information Systems Development, *Empirical Software Engineering*, Vol. 1, No. 3, pp. 201-240.

EL EMAM, K. AND MADHAVJI, N.H. (Eds.) (1999) *Elements of Software Process Assessment and Improvement*, Los Alamitos, California: IEEE Computer Society Press.

EL EMAM, K., DROUIN, J.-N., AND MELO, W. (Eds.) (1998) *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*, Los Alamitos, California: IEEE Computer Society Press.

EL EMAM, K., FUSARO, P., AND SMITH, B. (1999) Success Factors and Barriers for Software Process Improvement, in R. Messnarz and C. Tully (Eds.) *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press, pp. 355-371.

EL EMAM, K., GOLDENSON, D.R., MCCURLEY J., AND HERBSLEB, J. (2001) Modeling the Likelihood of Software Process Improvement: An Exploratory Study, *Empirical Software Engineering*, Vol. 6, No. 3, pp. 207-229.

ELDEN, M. AND CHISHOLM, R.F. (1993) Emerging Varieties of Action Research: Introduction to the Special Issue, *Human Relations*, Vol. 46, No. 2, pp. 121-142.

EMERY, M. AND PURSER, R.E. (1996) *The Search Conference*, San Francisco: Jossey-Bass.

FARAJ, S. AND SPROULL, L. (2000) Coordinating Expertise in Software Development Teams, *Management Science*, Vol. 46, No. 12, pp. 1554-1568.

FAYAD, M.E. AND LAITINEN, M. (1997) Process Assessment Considered Wasteful, *Communications of the ACM*, Vol. 40, No. 11, pp. 125-128.

FAYAD, M.E., LAITINEN, M., AND WARD, R.P. (2000) Software Engineering in the Small, *Communications of the ACM*, Vol. 43, No. 3, pp. 115-118.

FEIGENBAUM, A.V. (1991) *Total Quality Control*, Fortieth Anniversary Edition, New York: McGraw-Hill.

FEILER, P.H. AND HUMPHREY, W.S. (1993) Software Process Development and Enactment: Concepts and Definitions, *Proceedings of the Second International Conference on the Software Process*, IEEE Computer Society Press, pp. 28-40.

FENTON, N.E. (1991) *Software Metrics: A Rigorous Approach*, London: Chapman & Hall.

FENTON, N.E. (1994) Software Measurement: A Necessary Scientific Basis, *IEEE Transactions on Software Engineering*, Vol. 20, No. 3, pp. 199-206.

FENTON, N.E. (2001) Conducting and Presenting Empirical Software Engineering, *Empirical Software Engineering*, Vol. 6., No. 3, pp. 195-200.

FENTON, N.E. AND NEIL, M. (1999) Software Metrics: Successes, Failures and New Directions, *Journal of Systems and Software*, Vol. 47, No. 2-3, pp. 149-157.

FENTON, N.E. AND PFLEEGER, S.H. (1996) *Software Metrics: A Rigorous and Practical Approach*, London: International Thomson Computer Press.

FENTON, N.E., PFLEEGER, S.H., AND GLASS, R.L. (1994) Science and Substance: A Challenge to Software Engineers, *IEEE Software*, Vol. 11, No. 4, pp. 86-95.

FINK, A. AND KOSECOFF, J. (1998) *How to Conduct Surveys: A Step-By-Step Guide*, Second Edition, Thousand Oaks, California: Sage Publications.

FINKELSTEIN, A., KRAMER, J., AND NUSEIBEH, B. (1994) *Software Process Modelling and Technology*, Taunton, Somerset, U.K.: Research Studies Press.

FIOL, C.M. AND LYLES, M.A. (1985) Organizational Learning, *Academy of Management Review*, Vol. 10, No.4, pp. 803-813.

FISCHER, G. AND OSTWALD, J. (2001) Knowledge Management: Problems, Promises, Realities, and Challenges, *IEEE Intelligent Systems*, Vol. 16, No.1, pp. 60-72.

FISCHHOFF, B. (1975) Hindsight/ Foresight: The Effect of Outcome Knowledge on Judgment Under Uncertainty, *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 1, No. 3, pp. 288-99.

FITZGERALD, B. (1996) Formalized Systems Development Methodologies: A Critical Perspective, *Information Systems Journal*, Vol. 6, No. 1, pp. 3-23.

FLORAC, W.A. AND CARLETON, A.D. (1999) *Measuring the Software Process: Statistical Process Control for Software Process Improvement*, Reading, Massachusetts: Addison-Wesley.

FLORAC, W.A., PARK, R.E., AND CARLETON, A.D. (1997) *Practical Software Measurement: Measuring for Process Management and Improvement*, Guidebook, CMU/SEI-97-HB-003, Carnegie Mellon University, Software Engineering Institute.

FLOYD, C. (1992) Software Development as Reality Construction, in C. Floyd *et al.* (Eds.), *Software Development and Reality Construction*, Berlin: Springer-Verlag, pp. 86-100.

FLOYD, C., ZÜLLIGHOVEN, H., BUDDE, R., AND KEIL-SLAWIK, R. (Eds.) (1992) *Software Development and Reality Construction*, Berlin: Springer-Verlag.

FOWLER, P. AND RIFKIN, S. (1990) *Software Engineering Process Group Guide*, Technical Report CMU/SEI-90-TR-24, Carnegie Mellon University, Software Engineering Institute.

FRENCH, W.L. (1963) Processes vis-à-vis Systems: Toward a Model of the Enterprise and Administration, *Academy of Management Journal*, Vol. 6, March, pp. 46-58.

FRENCH, W.L. AND BELL, C.H. JR. (1999) *Organization Development: Behavioral Science Interventions for Organization Improvement*, Sixth Edition, Upper Saddle River, New Jersey: Prentice-Hall.

FUGGETTA, A. (1999) Rethinking the Modes of Software Engineering Research, *The Journal of Systems and Software*, Vol. 47, No. 2-3, pp.133-138.

FUGGETTA, A. (2000) Software Process: A Roadmap, in A. Finkelstein (Ed.), *The Future of Software Engineering*, New York: ACM Press, pp. 25-34.

FUSARO, P., EL EMAM, K., AND SMITH, B. (1998) *The Internal Consistency of the 1987 SEI Maturity Questionnaire and the SPICE Capability Dimension*, Empirical Software Engineering, Vol. 3, No. 2, pp. 179-201.

GABLE, G.G. (1994) Integrating Case Study and Survey Research Methods: An Example in Information Systems, *European Journal of Information Systems*, Vol. 3, No. 2, pp. 112-126.

GADAMER, H.G. (1989) *Truth and Method*, Second, Revised Edition, London: Sheed & Ward.

GAFFNEY, J., CRUICKSHANK, R., WERLING, R., AND FELBER, H. (1995) *The Software Measurement Guidebook*, Boston, Massachusetts: International Thomson Computer Press.

GALLIERS, R.D. (1992) Choosing Information Systems Research Approaches, in R.D. Galliers (Ed.), *Information Systems Research: Issues, Methods and Practical Guidelines*, Information systems series, Oxford: Blackwell, pp. 144-162.

GALLIERS, R.D. AND LAND, F.F. (1987) Choosing Appropriate Information-Systems Research Methodologies, *Communications of the ACM*, Vol. 30, No. 11, pp. 900-902.

GARVIN, D.A. (1983) Quality on the Line, *Harvard Business Review*, Vol. 61, No. 5, pp. 65-75.

GARVIN, D.A. (1984) Japanese Quality Management, *Columbia Journal of World Business*, Vol. 19, No. 3, pp. 3-19.

GARVIN, D.A. (1993) Building a Learning Organization, *Harvard Business Review*, Vol. 71, No.4, pp. 78-91.

GARVIN, D.A. (2000) *Learning in Action: A Guide to Putting the Learning Organization to Work*, Boston, Massachusetts: Harvard Business School Press.

GIBBS, W.W. (1994) Software's Chronic Crisis, *Scientific American*, Vol. 271, September, pp. 72-81.

GILB, T. (1976) *Software metrics*, Lund, Sweden: Studentlitteratur.

GINSBERG, M.P. AND QUINN, L.H. (1995) *Process Tailoring and the Software Capability Maturity Model, Technical Report*, CMU/SEI-94-TR-024, Carnegie Mellon University, Software Engineering Institute.

GIOIA, D. AND PITRE, E. (1990) Multiparadigm Perspectives on Theory Building, *Academy of Management Review*, Vol. 15, No. 4, pp. 584-602.

GJERSVIK, R. (1993) *The Construction of Information Systems in Organizations: An Action Research Project on Technology, Organizational Closure, Reflection and Change*, Doctoral Dissertation, Trondheim, Norway: Norwegian University of Science and Technology.

GLASS, R.L. (1994) The Software-Research Crisis, *IEEE Software*, Vol. 11, No. 6, pp. 42-47.

GLASS, R.L. (1995) *Software Creativity*, Englewood Cliffs, New Jersey: Prentice-Hall.

GLASS, R.L. (1997) Pilot Studies: What, Why, and How, *Journal of Systems and Software*, Vol. 36, No. 1, pp. 85-97.

GLASS, R.L. (1999) *Computing Calamities: Lessons Learned from Products, Projects, and Companies that Failed*, Upper Saddle River, New Jersey: Prentice-Hall.

GOLDENSON, D.R. AND HERBSLEB, J.D. (1995) *After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success*, Technical Report, CMU/SEI-95-TR-009, Carnegie Mellon University, Software Engineering Institute.

GOLDENSON, D.R., EL EMAM, K., HERBSLEB, J., AND DEEPHOUSE, C. (1999a) Empirical Studies of Software Process Assessment Methods, in K. El Emam and N.H. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement*, Los Alamitos, California: IEEE Computer Society Press, pp. 177-218.

GOLDENSON, D.R., GOPAL, A., AND MUKHOPADHYAY, T. (1999b) Determinants of Success in Software Measurement Programs: Initial Results, *Proceedings of the Sixth International Software Metrics Symposium*, IEEE Computer Society Press, pp. 10-21.

GOLES, T. AND HIRSCHHEIM, R. (2000) The Paradigm is Dead, the Paradigm is Dead … Long Live the Paradigm: The Legacy of Burrell and Morgan, *Omega: The International Journal of Management Science*, Vol. 28, No.3, pp. 249-268.

GOPALAKRISHNAN, S. AND DAMANPOUR, F. (1997) A Review of Innovation Research in Economics, Sociology and Technology Management, *Omega: International Journal of Management Science*, Vol. 25, No.1, pp. 15-28.

GRADY, R.B. (1992) *Practical Software Metrics for Project Management and Process Improvement*, Englewood Cliffs, New Jersey: Prentice-Hall.

GRADY, R.B. (1997) *Successful Software Process Improvement*, Upper Saddle River, New Jersey: Prentice-Hall.

GRADY, R.B. AND CASWELL, D. (1987) *Software Metrics: Establishing a Company-wide Program*, Englewood Cliffs, New Jersey: Prentice-Hall.

GRAY, E.M. AND SMITH, W.L. (1998) On the Limitations of Software Process Assessment and the Recognition of a Required Re-orientation for Global Process Improvement, *Software Quality Journal*, Vol. 7, No. 1, pp. 21-34.

GREENWOOD, D.J. AND LEVIN, M. (1998) *Introduction to Action Research: Social Research for Social Change*, Thousand Oaks, California: Sage.

GREENWOOD, D.J. AND LEVIN, M. (2000) Reconstructing the Relationships Between Universities and Society Through Action Research, in N.K. Denzin and Y.S. Lincoln (Eds.), *Handbook of Qualitative Research*, Second Edition, Thousand Oaks, California: Sage, pp. 85-106.

GREMBA, J. AND MYERS, C. (1997) The IDEAL Model: A Practical Guide for Improvement, Carnegie Mellon University, Software Engineering Institute, *Bridge*, Issue 3.

GRESSE, C., HOISL, B., AND WÜST, J. (1995) *A process Model for GQM-Based Measurement*, Technical Report, STTI-95-04-E, Software Technology Transfer Initiative, Kaiserslautern, Germany.

GROTH, L. (1999) *Future Organizational Design: The Scope for the IT-Based Enterprise*, Chichester: Wiley.

GROVER, V. (1999) From Business Reengineering to Business Process Change Management: A Longitudinal Study of Trends and Practices, *IEEE Transactions on Engineering Management*, Vol. 46, No. 1, pp. 36-46.

GUBA, E.G. (1987) What Have We Learned About Naturalistic Evaluation?, *Evaluation Practice*, Vol. 8, pp. 23-43.

GUBA, E.G. AND LINCOLN, Y.S. (1994) Competing Paradigms in Qualitative Research, in N.K. Denzin and Y.S. Lincoln (Eds.), *Handbook of Qualitative Research*, Thousand Oaks, California: Sage, pp. 105-117.

GUILFORD, J.P. (1954) *Psychometric Methods*, Second Edition, New York: McGraw-Hill.

GUINAN, P.J., COOPRIDER, J.G., AND FARAJ, S. (1998) Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach, *Information Systems Research*, Vol. 9, No. 2, pp. 101-125.

HABERMAS, J. (1984) *The Theory of Communicative Action: Reason and the Rationality of Society* (Translated by T. McCarthy), Boston: Beacon.

HAIR, J.P., ANDERSON, R.E., TATHAM, R.L., AND BLACK, W.C. (1998) *Multivariate Data Analysis*, Fifth edition, Upper Saddle River, New Jersey: Prentice-Hall.

HAMBRICK, D.C., GELETKANYCZ, M.A., AND FREDRICKSON, J.W. (1993) Top Executive Commitment to the Status Quo: Some Tests for its Determinants, *Strategic Management Journal*, Vol. 14, No. 6, pp. 401-418.

HAMMER, M. (1990) Reengineering Work: Don't Automate – Obliterate, *Harvard Business Review*, Vol. 68, No. 4, pp. 104-112.

HAMMER, M. (1996) *Beyond Reengineering: How the Process-Centered Organization is Changing Our Work and Our Lives*, London: HarperCollins.

HAMMER, M. AND CHAMPY, J. (1993) *Reengineering the Corporation: A Manifesto for Business Revolution*, New York: Harper Business.

HAND, D.J. (1996) Statistics and the Theory of Measurement, *Journal of the Royal Statistics Society*, Series A, Vol. 159, Part 3, pp. 445-473.

HARRISON, R., BADOO, N., BARRY, E., BIFFL, S., PARRA, A., WINTER, B., AND WUEST, J. (1999) Directions and Methodologies for Empirical Software Engineering Research, *Empirical Software Engineering*, Vol. 4, No. 4, pp. 405-410.

HASSARD, J. (1988) Overcoming Hermeticism in Organization Theory: An Alternative to Paradigm Incommensurability, *Human Relations*, Vol. 41, No. 3, pp. 247-259.

HAYS, W.L. (1994) *Statistics*, Fifth edition, New York: Harcourt Brace.

HEMPEL, C.G. (1966) *Philosophy of Natural Science*, Foundations of Philosophy Series, Englewood Cliffs, New Jersey: Prentice-Hall.

HENDERSON, J.C. AND LEE, S. (1992) Managing IS Design Teams: A Control Theories Perspective, *Management science*, Vol. 38, No. 6, pp. 757-777.

HERBSLEB, J.D. (1998) Hard Problems and Hard Science: On the Practical Limits of Experimentation, *IEEE TCSE Software Process Newsletter*, No. 11, pp. 18-21.

HERBSLEB, J.D. AND GOLDENSON, D.R. (1996) A Systematic Survey of CMM Experience and Results, *Proceedings of the Eighteenth International Conference on Software Engineering*, IEEE Computer Society Press, pp. 323-330.

HESSELBEIN, F., GOLDSMITH, M., AND BECKHARD, R. (Eds.) (1996) *The Leader of the Future: New Visions, Strategies, and Practices for the Next Era*, San Francisco: Jossey-Bass.

HOARE, C.A.R. (1984) Programming: Sorcery or Science?, *IEEE Software*, Vol. 1, No. 2, pp. 5-16.

HOCH, D.J., ROEDING, C.R., PURKERT, G., AND LINDNER, S.K. (2000) *Secrets of Software Success: Management Insights from 100 Software Firms around the World*, Boston, Massachusetts: Harvard Business School Press.

HOHMANN, L. (1997) *Journey of the Software Professional: A Sociology of Software Development*, Upper Saddle River, New Jersey: Prentice Hall.

HOWARD, A. (2001) Software Engineering Project Management, *Communications of the ACM*, Vol. 44, No. 5, pp. 23-24.

HUBER, G.P. (1991) Organizational Learning: the Contributing Processes and the Literatures, *Organization Science*, Vol. 2, No. 1, pp. 88-115.

HUMPHREY, W.S. (1989) *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley.

HUMPHREY, W.S. (1997) *Managing Technical People: Innovation, Teamwork, and the Software Process*. Reading, Massachusetts: Addison-Wesley.

HUMPHREY, W.S., SNYDER, T. AND WILLIS, R. (1991) Software Process Improvement at Hughes Aircraft, *IEEE Software*, Vol. 8, No. 4, pp. 11-23.

HUNT, R. AND BUZAN, T. (1999) *Creating a Thinking Organization: Groundrules for Success*, Hampshire, England: Gower.

HUNTER, J.E. AND SCHMIDT, F.L. (1990) *Methods of Meta-Analysis: Correcting Error and Bias in Research Findings*, Newbury Park, California: Sage Publications.

HUNTER, R.B. AND THAYER, R.H (Eds.) (2001) *Software Process Improvement*, Los Alamitos, California: IEEE Computer Society Press.

HYMAN, R.B. (1993) Creative Chaos in High-Performance Teams: An Experience Report, *Communications of the ACM*, Vol. 36, No. 10, pp. 57-60.

IEEE Std 610.12-1990 *IEEE Standard Glossary of Software Engineering Terminology*.

IEEE Std 983-1986 *IEEE Guide for Software Quality Assurance Planning*.

ILSTAD, S. (1997) *A system for Appraising Questionnaires* (in Norwegian), Working Paper, No. 7, Norwegian University of Science and Technology.

IMAI, M. (1986) *Kaizen: The Key to Japan's Competitive Success*, New York: McGraw-Hill.

ISENBERG, D.J. (1984) How Senior Managers Think, *Harvard Business Review*, Vol. 62, No. 2, pp. 81-90.

ISHIKAWA, K. (1986) *Guide to Quality Control*, Second Edition, New York: Quality Resources.

ISHIKAWA, K. (1990) *Introduction to Quality Control*, London: Chapman & Hall.

ISO 8402:1994 *Quality Management and Quality Assurance – Quality Vocabulary*, International Organization for Standardization.

ISO 9000:2000 *Quality Management Systems – Fundamentals and vocabulary*, International Organization for Standardization.

ISO 9001:2000 *Quality Management Systems – Requirements*, International Organization for Standardization.

ISO/IEC 12207: 1995 *Information technology – Software Life Cycle Processes*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC CD 15504:2001 *Software Engineering – Process Assessment* (Parts 1 to 5), International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC CD 15504-2:2001 *Software Engineering – Process Assessment – Part 2: Performing an Assessment*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC CD 15939:2001 *Software Engineering – Software Measurement Process Framework*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC TR 15504-1:1998 *Information Technology – Software Process Assessment – Part 1: Concepts and Introductory Guide*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC TR 15504-7:1998 *Information Technology – Software Process Assessment – Part 7: Guide for Use in Process Improvement*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC TR 15504-8:1998 *Information Technology – Software Process Assessment – Part 8: Guide for Use in Determining Supplier Process Capability*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC TR 15504-9:1998 *Information Technology – Software Process Assessment – Part 9: Vocabulary*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC WD 15504-4:2001 *Software Engineering – Process Assessment – Part 4: Guidance on use for Process Improvement and Process Capability Determination*, International Organization for Standardization and the International Electrotechnical Commission.

ISO/IEC WD 9000-3:2001 *Software and System Engineering – Guidelines for the Application of ISO 9000:2000 to Software*, International Organization for Standardization and the International Electrotechnical Commission.

JANSEN, P. AND SANDERS, J. (1998) Guidelines for Process Improvement, in K. El Emam, J.-N. Drouin and W. Melo (Eds.), *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*, Los Alamitos, California: IEEE Computer Society Press, pp. 171-192.

JEFFERY, D.R. AND VOTTA, L.G. (1999) Guest Editor's Special Section Introduction, *IEEE Transactions on Software Engineering*, Vol. 25, No. 4, pp. 435-437.

JELETIC, K., PAJERSKI, R., AND BROWN, C. (1996) *Software Process Improvement Guidebook*, Revision 1, Software Engineering Laboratory Series, SEL-95-102, Greenbelt, Maryland: Goddard Space Flight Center.

JELINEK, J.J. AND SCHOONHOVEN, C.B. (1990) *The Innovation Marathon*, Cambridge, UK: Basil-Blackwell.

JICK, T.D. (1979) Mixing Qualitative and Quantitative Methods: Triangulation in Action, *Administrative Science Quarterly*, Vol. 24, No. 4, pp. 602-611.

JOHNSON, D.L. AND BRODMAN, J.G. (1999) Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects, in K. El Emam and N.H. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement*, Los Alamitos, California: IEEE Computer Society Press, pp. 239-257.

JOHNSON-LAIRD, P.N. (1983) *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*, Cambridge: Cambridge University Press.

JONES, C. (1999) The Economics of Software Process Improvement, in K. El Emam and N.H. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement*, Los Alamitos, California: IEEE Computer Society Press, pp. 133-149.

JOY, B. AND KENNEDY, K. (Eds.) (1999) *Information Technology Research: Investing in Our Future*, President's Information Technology Advisory Committee (PITAC), Report to the

President, Arlington, VA: National Coordination Office for Information Technology Research and Development.

JURAN, J.M. (1964) *Managerial Breakthrough: A New Concept of the Manager's Job*, New York: McGraw-Hill.

JURAN, J.M. (1992) *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services*, New York: Free Press.

JURAN, J.M. AND GODFREY, A.B. (Eds.) (1999) *Juran's Quality Handbook*, Fifth Edition, New York: McGraw-Hill.

JØRGENSEN, M., SJØBERG, D., AND CONRADI, R (1998) Reuse of Software Development Experience at Telenor Telecom Software, *Proceedings of the European Software Process Improvement Conference (EuroSPI'98)*, Gothenburg, Sweden, November 16-18.

KAISER, H.F. (1960) The Application of Electronic Computers to Factor Analysis, *Educational and Psychological Measurements*, Vol. 20, pp. 141-151.

KAISER, H.F. (1970) A Second Generation Little Jiffy, *Psychometrika*, Vol. 35, pp. 401-417.

KANO, N., NOBUHIRO, S., TAKAHASHI, F., AND TSUJI, S. (1984) Attractive Quality and Must Be Quality, *Quality Magazine*, Vol. 14, No. 2, pp. 39-48.

KANUK, L. AND BERENSON, C. (1975) Mail Surveys and Response Rates: A Literature Review, *Journal of Marketing Research*, Vol. 12, pp. 440-453.

KAPLAN, A. (1964) *The Conduct of Inquiry: Methodology for Behavioral Science*, San Francisco: Chandler.

KAPLAN, D. (1990) Evaluating and Modifying Covariance Structure Models: A Review and Recommendation, *Multivariate Behavioral Research*, Vol. 25, No. 2, pp. 137-155.

KAPLAN, R.S. AND NORTON, D.P. (1996) *The Balanced Scorecard: Translating Strategy into Action*, Boston, Massachusetts: Harvard Business School Press.

KAPLAN, R.S. AND NORTON, D.P. (2000) *The Strategy-focused Organization: How Balanced Scorecard Companies Thrive in the New Business Environment*, Boston, Massachusetts: Harvard Business School Press.

KARLSSON, E.-A. (Ed.) (1995) *Software Reuse: A Holistic Approach*, Chichester: John Wiley & Sons.

KATZ, D. AND KAHN, R.L. (1978) *The Social Psychology of Organizations*, Second Edition, New York: Wiley.

KAUTZ, K. AND LARSEN, E.Å. (2000) Diffusion Theory and Practice: Disseminating Quality Management and Software Process Improvement Innovations, *Information Technology & People*, Vol. 13, No. 1, pp. 11-26.

KAWALEK, P. AND WASTELL, D.G. (1999) Software Development and Organizational Viability: An Account of the Impact of Organizational Issues Upon Software Quality, in E. McGuire (Ed.), *Software Process Improvement: Concepts and Practices*, London: Idea Group Publishing, pp. 60-75.

KELSEY, R.B. (1999) *Chaos and Complexity in Software: Challenging the Industry and the New Science*, New York: Nova Science Publishers.

KERLINGER, F. (1986) *Foundations of Behavioral Research*, New York: Holt, Rinehart and Winston.

KIM, D.H. (1993) The Link between Individual and Organizational Learning, *Sloan Management Review*, Vol. 35, No. 1, pp. 37-50.

KITCHENHAM, B.A. (1996) *Software Metrics: Measurement for Software Process Improvement*, Oxford, UK: NCC Blackwell.

KITCHENHAM, B.A. (1996/1998) Evaluating Software Engineering Methods and Tools, Parts 1 to 12, *SIGSOFT Software Engineering Notes*, Vols. 21-23.

KITCHENHAM, B.A., PFLEEGER, S.L., PICKARD, L.M., JONES, P.W., HOAGLIN, D.C., EL EMAM, K., AND ROSENBERG, J. (Forthcoming) Preliminary guidelines for empirical research in software engineering, *IEEE Transactions on Software Engineering*.

KITCHENHAM, B.A., PICKARD, L.M., AND PFLEEGER, S.L. (1995) Case Studies for Method and Tool Evaluation, *IEEE Software*, Vol. 12, No. 4, pp. 52-62.

KLEIN, H.K. AND MYERS, M.D. (1999) A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, *MIS Quarterly*, Vol. 23, No. 1, pp. 67-93.

KLINE, S.J. (1985) Innovation Is Not a Linear Process, *Res. Policy*, Vol. 14, No. 4, pp. 36-54.

KOLB, D.A. (1984) *Experiential Learning: Experience as the Source of Learning and Development*, Englewood Cliffs, New Jersey: Prentice-Hall.

KONISHI, S. (1981) Normalizing Transformations of some Statistics in Multivariate Analysis, *Biometrika*, Vol. 68, No. 3, pp. 647-651.

KOTLER, P. (1988) *Marketing Management: Analysis, Planning, Implementation, and Control*, Englewood Cliffs, New Jersey: Prentice-Hall.

KOTTER, J.P. (1996) *Leading Change*, Boston, Massachusetts: Harvard Business School Press.

KRASNER, H. (1999) The Payoff for Software Process Improvement: What it is and How to Get it, in K. El Emam and N.H. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement*, Los Alamitos, California: IEEE Computer Society Press, pp. 151-176.

KRAUT, A.I. (Ed.) (1996) *Organizational Surveys: Tools for Assessment and Change*, San Francisco: Jossey-Bass.

KRAUT, R.E. AND STREETER, L. (1995) Coordination in Software Development, *Communications of the ACM*, Vol. 38, No. 3, pp. 69-81.

KROGSTIE, J. (1995) *Conceptual Modeling for Computerized Information Systems Support in Organizations*, Doctoral Dissertation, Trondheim, Norway: Norwegian University of Science and Technology.

KROGSTIE, J. (1996) Use of Methods and CASE-Tools in Norway: Results from a Survey, *Automated Software Engineering*, Vol. 3, Nos. 3/4, pp. 347-367.

KROSNICK, J.A. (1999) Survey Research, *Annual Review of Psychology*, Vol. 50, pp. 537-567.

KRUCHTEN, P. (2000) *The Rational Unified Process: An Introduction*, Second Edition, Reading, Massachusetts: Addison-Wesley.

KUHN, T.S. (1996) *The Structure of Scientific Revolutions*, Third Edition, Chicago: The University of Chicago Press.

KUNDA, G. (1992) *Engineering Culture: Control and Commitment in a High-Tech Corporation*, Philadelphia: Temple University Press.

KUVAJA, P., SIMILA, J., KRZANIK, L., BICEGO, A., SOUKKONEN, S., AND KOCH, G. (1994) *Software Process Assessment and Improvement: The BOOTSTRAP Approach*, Blackwell.

LAITINEN, M., FAYAD, M., AND WARD, R. (2000) Software Engineering in the Small, *IEEE Software*, Vol. 17, No. 5, pp. 75-77.

LANT, T.K. AND MEZIAS, S.J. (1992) An Organizational Learning Model of Convergence and Reorientation, *Organization Science*, Vol. 3, No. 1, pp. 47-71.

LARSEN, E.Å, AND KAUTZ, K. (1997) Quality Assurance and Software Process Improvement in Norway, *Software Process – Improvement and Practice*, Vol.3, No. 2, pp. 71-86.

LATOUR, B. (1987) *Science in Action: How to Follow Scientists and Engineers through Society*, Cambridge, Massachusetts: Harvard University Press.

LAU, F. (1999) Toward a Framework for Action Research in Information Systems Studies, *Information Technology & People*, Vol. 12, No. 2, pp. 148-175.

LAVE, J. AND WENGER, E. (1991) *Situated Learning: Legitimate Peripheral Participation*, Cambridge: Cambridge University Press.

LAWLER, E.E. III (1997) Rethinking Organization Size, *Organizational Dynamics*, Vol. 26, No. 2, pp. 24-35.

LAWRENCE, P.R. AND LORSCH, J.W. (1986) *Organization and Environment: Managing Differentiation and Integration*, Second Edition, Boston: Harvard Business School Press.

LEE, A.S. (1989) A Scientific Methodology for MIS Case Studies, *MIS Quarterly*, Vol. 13, No. 1, pp. 33-50.

LEE, A.S. (1991) Integrating Positivist and Interpretive Approaches to Organizational Research, *Organization Science*, Vol. 2, No. 4, pp. 342-365.

LEE, A.S. (1999a) Researching MIS, in W.L. Currie and R.D. Galliers (Eds.), *Rethinking Management Information Systems: An Interdisciplinary Perspective*, Oxford: Oxford University Press, pp. 7-27.

LEE, T.W. (1999b) *Using Qualitative Methods in Organizational Research*, Thousand Oaks, California: Sage.

LEHMAN, M.M. (1989) Uncertainty in Computer Applications and its Control through the Engineering of Software, *Software Maintenance: Research and Practice*, Vol. 1, No. 1, pp. 3-27.

LEIFER, R., MCDERMOTT, C.M., O'CONNOR, G.C., PETERS, L.S., RICE, M.P., AND VERYZER, R.W., JR. (2001) *Radical Innovation: How Mature Companies Can Outsmart Upstarts*, Boston, Massachusetts: Harvard Business School Press.

LEONARD-BARTON, D. (1995) *Wellsprings of Knowledge: Building and Sustaining the Sources of Innovation*, Boston: Harvard Business School Press.

LEVIN, M. (1997) Technology Transfer is Organizational Development: An Investigation into the Relationship between Technology Transfer and Organizational Change, *International Journal of Technology Management*, Vol. 14, Nos. 2/3/4, pp. 297-308.

LEVINTHAL, D.A. AND MARCH, J.G. (1993) The Myopia of Learning, *Strategic Management Journal*, Vol. 14, pp. 95-112.

LÉVI-STRAUSS, C. (1966) *The Savage Mind*, Chicago: University of Chicago Press.

LEVITT, B. AND MARCH, J.G. (1988) Organizational Learning, *Annual Review of Sociology*, Vol. 14, pp. 319-340.

LEWIN, K. (1951) *Field Theory in Social Sciences*, New York: Harper and Row.

LIED, H.J. AND STÅLHANE, T. (1999) Experience from Process Improvement in a SME, *Proceedings of the European Software Process Improvement Conference (EuroSPI'99)*, Pori, Finland, 25-27 October.

LIKERT, R. (1932) A Technique for the Measurement of Attitudes, *Archives of Psychology*, Vol. 22, No. 140.

LIKERT, R. (1967) *The Human Organization: Its Management and Value*, New York: McGraw-Hill.

LIKERT, R. AND ROSLOW, S. (1934) *The Effects Upon the Reliability of Attitude Scales of Using Three, Five or Seven Alternatives*, Working Paper, New York University.

LIPPITT, R., WATSON, J., AND WESTLY, B. (1958) *The Dynamics of Planned Change*, New York: Harcourt, Brace and World.

LISSITZ, R.W. AND GREEN, S.B. (1975) Effects of the Number of Scale Points on Reliability: A Monte Carlo Approach, *Journal of Applied Psychology*, Vol. 60, February, pp. 10-13.

LUCAS, H.C. (1975) *Why Information Systems Fail*, New York: Columbia University Press.

LYNCH, R.L. AND CROSS, K.C. (1991) *Measure Up! Yardstick for Continuous Improvement*, Cambridge, Massachusetts: Blackwell Business.

LYNN, G.S., MORONE, J.G., AND PAULSON, A.S. (1996) Marketing and Discontinuous Innovation: The Probe and Learn Process, *California Management Review*, Vol. 38, No. 3, pp. 8-37.

LYSGAARD, S. (1961) *Arbeiderkollektivet* (The Worker Collective), Oslo: Universitetsforlaget (in Norwegian).

MACKENZIE, D. AND WAJCMAN, J. (Eds.) (1985) *The Social Shaping of Technology*, Milton Keynes: Open University Press.

MADHAVJI, N.H. (1991) The Process Cycle, *Software Engineering Journal*, Vol. 6, No. 5, pp. 234-242.

MARCH, J.G. (1991) Exploration and Exploitation in Organizational Learning, *Organization Science*, Vol. 2, No. 1, pp. 71-87.

MARCH, J.G. (1999) *The Pursuit of Organizational Intelligence*, Malden, Massachusetts: Blackwell.

MARCH, J.G. AND OLSEN, J.P. (1976) *Ambiguity and Choice in Organizations*, Bergen, Norway: Universitetsforlaget.

MARCH, J.G. AND SIMON, H.A. (1958) *Organizations*, New York: John Wiley.

MARCH, J.G. AND SIMON, H.A. (1993) *Organizations*, Second Edition, Oxford, UK: Blackwell.

MARCH, J.G. AND SUTTON, R.I. (1997) Organizational Performance as a Dependent Variable, *Organization Science*, Vol. 8, No. 6, pp. 698-706.

MARQUIS, D.G. (1988) The Anatomy of Successful Innovations, in M.L. Tushman, and W.L. Moore (Eds.), *Readings in the Management of Innovation*, Second Edition, New York: HarperBusiness, pp. 79-87.

MARTIN, J. (1992) *Cultures in Organizations: Three Perspectives*, New York: Oxford University Press.

MATHIASSEN, L. (1998) *Reflective Systems Development*, Dr. Techn. Thesis, Aalborg University, Denmark.

MATHIASSEN, L. AND STAGE, J. (1992) The Principle of Limited Reduction in Software Design, *Information Technology & People*, Vol. 6, Nos. 2-3, pp. 171-185.

MATURANA, H.R. AND VARELA, F.J. (1980) *Autopoiesis and Cognition: The Realization of the Living*, London: Reidl.

MAYO, E. (1933) *The Human Problems of an Industrial Civilization*, Boston: Harvard University Press.

MAYO, E. (1945) *The Social Problems of an Industrial Civilization*, Boston: Harvard University Press.

MCFEELEY, B. (1996) *IDEAL: A User's Guide for Software Process Improvement*, Handbook, CMU/SEI-96-HB-01, Carnegie Mellon University, Software Engineering Institute.

MCGARRY, J. (Ed.) (1998) *Practical Software Measurement: A Foundation for Objective Project Management*, Version 3.1a, Office of the Under Secretary of Defense for Acquisition and Technology, Joint Logistics Commanders Joint Group on Systems Engineering.

MCIVER, J.P. AND CARMINES, E.G. (1981) *Unidimensional Scaling*, Beverly Hills: Sage Publications.

MEHNER, T. (1999) Siemens Process Assessment Approach, in R. Messnarz and C. Tully (Eds.), *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press, pp. 199-212.

MERRIAM-WEBSTER (1998) *Merriam-Webster's Collegiate Dictionary*, 10th Edition, Springfield, Massachusetts: Merriam-Webster.

MESSNARZ, R. (1999a) Software Process and Analysis: Concepts and Definitions, in R. Messnarz and C. Tully (Eds.) *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press, pp. 29-50.

MESSNARZ, R. (1999b) Summary and Outlook, in R. Messnarz and C. Tully (Eds.) *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press, pp. 389-393.

MESSNARZ, R. AND TULLY, C. (Eds.) (1999) *Better Software Practice for Business Benefit: Principles and Experience*, Los Alamitos, California: IEEE Computer Society Press.

MILLER, G.A. (1956) The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, *Psychological Review*, Vol. 63, pp.81-97.

MILLER, J., DALY, J., WOOD, M., ROPER, M., AND BROOKS, A. (1997) Statistical Power and its Subcomponents – Missing and Misunderstood Concepts in Empirical Software Engineering Research, *Information and Software Technology*, Vol. 39, No. 4, pp. 285-295.

MINTZBERG, H. (1979) An Emerging Strategy of "Direct" Research, *Administrative Science Quarterly*, Vol. 24, pp. 580-589.

MINTZBERG, H. (1983) *Structure in Fives: Designing Effective Organizations*, Englewood Cliffs, New Jersey: Prentice Hall.

MINTZBERG, H. (1989) *Mintzberg on Management: Inside Our Strange World of Organizations*, New York: The Free Press.

MINZTBERG, H. (1994) The Fall and Rise of Strategic Planning, *Harvard Business Review*, Vol. 72, No. 1, pp. 107-114.

MONTANGERO, C. (1999) The Software Process: Modelling and Technology, in J.-C. Derniame, B.A. Kaba, and D.G. Wastell (Eds.), *Software Process: Principles, Methodology, and Technology*, Lecture Notes in Computer Science, Vol. 1500, Berlin: Springer-Verlag, pp. 1-13.

MOORMAN, C. AND MINER, A.S. (1998) The Convergence of Planning and Execution: Improvisation in New Product Development, *Journal of Marketing*, Vol. 62, No. 3, July, pp. 1-20.

MORGAN, G. (1997) *Images of Organization*, Newbury Hill: Sage Publications.

MUMFORD, E. (1983) *Designing Human Systems for New Technology: The ETHICS Method*, Manchester: Manchester Business School.

MÖLLER, K.H. AND PAULISH, D.J. (1993) *Software Metrics: A Practitioner's Guide to Improved Product Development*, Los Alamitos, California: IEEE Computer Society Press.

NADLER, D.A. (1977) *Feedback and Organization Development: Using Data-based Methods*, Reading, Massachusetts: Addison-Wesley.

NAUR, P. (1983) Program Development Studies Based on Diaries, in T. Green, S. Payne, and G. van der Veer (Eds.), *Psychology of Computer Use*, London: Academic Press, pp. 159-170.

NAUR, P. AND RANDELL, B. (Eds.) (1969) Software Engineering, *Proceedings of the NATO Conference in Garmisch-Partenkirchen*, NATO Science Committee, Scientific Affairs Division, NATO, Brussels, January.

NEFF, F.W. (1966) Survey Research: A Tool for Problem Diagnosis and Improvement in Organizations, in A.W. Gouldner and S.M. Miller (Eds.), *Applied Sociology*, New York: Free Press, pp. 23-38.

NEUMAN, W.L. (2000) *Social Research Methods: Qualitative and Quantitative Approaches*, Fourth Edition, Boston: Allyn and Bacon.

NEVIS, E.C., DIBELLA, A.J., AND GOULD, J.M. (1995) Understanding Organizations as Learning Systems, *Sloan Management Review*, Vol. 37, Winter, pp. 73-85.

NEWMAN, M. AND ROBEY, D. (1992) A Social Process Model of User Analyst Relationships, *MIS Quarterly*, Vol. 16, No. 2, pp. 249-266.

NICOLINI, D. AND MEZNAR, M.B. (1995) The Social Construction of Organizational Learning: Conceptual and Practical Issues in the Field, *Human Relations*, Vol. 48, No.7, pp. 727-746.

NOLAN, A.J. (1999) Learning from Success, *IEEE Software*, Vol. 16, No. 1, pp. 97-105.

NONAKA, I. (1991) The Knowledge-Creating Company, *Harvard Business Review*, Vol. 69, No. 6, November-December, pp. 96-104.

NONAKA, I. (1994) A Dynamic Theory of Organizational Knowledge Creation, *Organization Science*, Vol. 5, No.1, pp.14-37.

NONAKA, I. AND KONNO, N. (1998) The Concept of "Ba": Building a Foundation for Knowledge Creation, *California Management Review*, Vol. 40, No. 3, pp. 40-54.

NONAKA, I. AND TAKEUCHI, H. (1995) *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, New York: Oxford University Press.

NOVICK, M. AND LEWIS, G. (1967) Coefficient Alpha and the Reliability of Composite Measurements, *Psychometrika*, Vol. 32, pp. 1-13.

NUNNALLY, J.C. (1978) *Psychometric Theory*, Second Edition, New York: McGraw-Hill.

NUNNALLY, J.C. AND BERNSTEIN, I.A. (1994) *Psychometric Theory*, Third Edition, New York: McGraw-Hill.

O'NEILL, P. AND SOHAL, A.S. (1999) Business Process Reengineering: A Review of Recent Literature, *Technovation*, Vol. 19, No. 9, pp. 571-581.

OLSON, T., HUMPHREY, W.S., AND KITSON, D. (1989) *Conducting SEI-assisted Software Process Assessments*, Technical Report, CMU/SEI-89-TR-07, Carnegie Mellon University, Software Engineering Institute.

OMAN, P. AND PFLEEGER, S.L. (1997) *Applying Software Metrics*, Los Alamitos, California: IEEE Computer Society Press.

ORLIKOWSKI, W.J. AND HOFMAN, J.D. (1997) An Improvisational Model for Change Management: The Case of Groupware Technologies, *Sloan Management Review*, Vol. 38, No. 2, pp. 11-21.

ORR, J.E. (1990) Sharing Knowledge, Celebrating Identity: Community Memory in a Service Culture, in D. Middleton and D. Edwards (Eds.) *Collective Remembering*, London: Sage, pp. 168-189.

ORR, J.E. (1996) *Talking about Machines: An Ethnography of a Modern Job*, Ithaca, New York: Cornell University Press.

OSBORN, A.F. (1963) *Applied Imagination: Principles and Procedures of Creative Problem Solving*, Third Edition, New York: Scribners.

OULD, M.A. (1996) CMM and ISO 9001, *Software Process – Improvement and Practice*, Vol. 2, pp. 281-289.

PARK, R.E., GOETHERT, W.B., AND FLORAC, W.A. (1996) *Goal-Driven Software Measurement – A Guidebook*, Handbook, CMU/SEI-96-HB-002, Carnegie Mellon University, Software Engineering Institute.

PARNAS, D.L. AND CLEMENTS, P.C. (1986) A Rational Design Process: How and Why to Fake It, *IEEE Transactions on Software Engineering*, Vol. 12, No. 2, pp. 251-257.

PARNES, S.J. (1967) *Creative Behavior Guidebook*, New York: Scribners.

PATTON, M.Q. (1990) *Qualitative Evaluation and Research Methods*, Second Edition, Newbury Park, California: Sage.

PAULK, M.C. AND KONRAD, M. (1994) Measuring Process Capability versus Organizational Process Maturity, *Proceedings of the 4th International Conference on Software Quality*, October.

PAULK, M.C., WEBER, C.V., AND CHRISSIS, M.B. (1999) The Capability Maturity Model for Software, in K. El Emam and N.H. Madhavji (Eds.), *Elements of Software Process*

Assessment and Improvement, Los Alamitos, California: IEEE Computer Society Press, pp. 3-22.

PAULK, M.C., WEBER, C.V., CURTIS, B., AND CHRISSIS, M.B. (1995) *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, Massachusetts: Addison-Wesley.

PAYNE, S. (1951) *The Art of Asking Questions*, Princeton, New Jersey: Princeton University Press.

PETERS, T.J. (1988) *Thriving on Chaos: Handbook for a Management Revolution*, New York : Harper & Row.

PETERS, T.J. (1992) *Liberation Management: Necessary Disorganization for the Nanosecond Nineties*, New York: Alfread A. Knopf.

PETERS, T.J. AND WATERMAN, R.H. (1982) *In Search of Excellence: Lessons from America's Best-Run Companies*, New York: Harper & Row.

PETERSON, B. (1995) Software Engineering Institute, *Software Process Improvement and Practice*, Pilot Issue, John Wiley & Sons Ltd., August.

PFEFFER, J. AND SUTTON, R.I. (2000) *The Knowing-Doing Gap: How Smart Companies Turn Knowledge into Action*, Boston, Massachusetts: Harvard Business School Press.

PFLEEGER, S.L. (1994/1995) Experimental Design and Analysis in Software Engineering, Parts 1 to 5, *SIGSOFT Software Engineering Notes*, Vols. 19-20.

PFLEEGER, S.L. (1997) Guidelines for Applying Research Results, *IEEE Software*, Vol. 14, No. 3, pp. 102-104.

PFLEEGER, S.L. (1999a) Albert Einstein and Empirical Software Engineering, *IEEE Computer*, Vol. 38, No. 10, pp.32-38.

PFLEEGER, S.L. (1999b) Understanding and Improving Technology Transfer in Software Engineering, *Journal of Systems and Software*, Vol. 47, No. 2-3, pp. 111-124.

PFLEEGER, S.L. AND MENEZES, W. (2000) Marketing Technology to Software Practitioners, *IEEE Software*, Vol. 17, No. 1, pp. 27-33.

PIAGET, J. (1970) *Genetic Epistemology*, New York: Columbia University Press.

PINSONNEAULT, A. AND KRAEMER, K.L. (1993) Survey Research Methodology in Management Information Systems: An Assessment, *Journal of Management Information Systems*, Vol. 10, No. 2, pp.75-105.

POLANYI, M. (1966) *The Tacit Dimension*, New York: Doubleday.

POOLE, M.S., VAN DE VEN, A.H., DOOLEY, K., AND HOLMES, M.E. (2000) *Organizational Change and Innovation Processes: Theory and Methods for Research*, Oxford: Oxford University Press.

POPPER, K.R. (1959) *The Logic of Scientific Discovery*, London: Hutchinson.

POPPER, K.R. (1979) *Objective Knowledge: An Evolutionary Approach*, Revised Edition, Oxford: Clarendon Press.

POTTS, C. (1993) Software-Engineering Research Revisited, *IEEE Software*, Vol. 10, No. 5, pp. 19-28.

POWELL, T.C. (1995) Total Quality Management as Competitive Advantage: A Review and Empirical Study, *Strategic Management Journal*, Vol. 16, No. 1, pp. 15-37.

PRAHALAD, C.K. AND HAMEL, G. (1990) The Core Competence of the Corporation, *Harvard Business Review*, Vol. 68, No. 3, pp. 79-91.

PRESLEY, A. SARKIS, J., AND LILES, D.H. (2000) A Soft-Systems Methodology Approach for Product and Process Innovation, *IEEE Transactions on Engineering Management*, Vol. 47, No. 3, pp. 379-392.

PRESSMAN, R.S. (1988) *Making Software Engineering Happen: A Guide for Instituting the Technology*, Englewood Cliffs, New Jersey: Prentice Hall.

PROFES (1999) *PROFES User Manual*, Final Version, http://www.iese.fhg.de/Profes.

PULFORD, K., KUNTZMANN-COMBELLES, A., AND SHIRLAW, S. (1996) *A Quantitative Approach to Software Management: The* ***ami*** *Handbook*. Wokingham, England: Addison-Wesley.

QUINN, J.B. (1985) Managing Innovation: Controlled Chaos, *Harvard Business Review*, Vol. 63, No. 3, pp. 73-84.

RADEMACHER, R.A. (1999) Statistical Power in Information Systems Research: Application and Impact on the Discipline, *Journal of Computer Information Systems*, Vol. 39, No. 4, pp. 1-7.

RAGHAVAN, S.A. AND CHAND, D.R. (1989) Diffusing Software-Engineering Methods, *IEEE Software*, Vol. 6, No. 4, pp. 81-90.

RAYMOND, E.S. (1999) *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Sebastopol, California: O'Reilly.

RAYMOND, L., BERGERON, F., AND RIVARD, S. (1998) Determinants of Business Process Reengineering Success in Small and Large Enterprises: An Empirical Study in the Canadian Context, *Journal of Small Business Management*, Vol. 36, No. 1, pp. 72-85.

REICHARDT, C.S. AND RALLIS, S.F. (Eds.) (1994) *The Qualitative-Quantitative Debate: New Perspectives*, San Francisco: Jossey-Bass.

RETTIG, M. AND SIMONS, G. (1993) A Project Planning and Development Process for Small Teams, *Communications of the ACM*, Vol. 36, No. 10, pp. 45-55.

RIFKIN, S. (2001) What Makes Measuring Software So Hard?, *IEEE Software*, Vol. 18, No. 3, pp. 41-45.

ROBINSON, H., HALL, P., HOVENDEN, F., AND RACHEL, J. (1998) Postmodern Software Development, *The Computer Journal*, Vol. 41, No. 6, pp. 363-375.

ROETZHEIM, W.H. (1988) *Structured Computer Project Management*, Englewood Cliffs, New Jersey: Prentice Hall.

ROGERS, E.M. (1995) *Diffusion of Innovations*, Fourth Edition, New York: The Free Press.

ROOIJMANS, J., AERTS, H., AND VAN GENUCHTEN, M. (1996) Software Quality in Consumer Electronics Products, *IEEE Software*, Vol. 13, No. 1, pp. 55-64.

ROSEN, S.M. (1994) *Science, Paradox, and the Moebius Principle: The Evolution of a "Transcultural" Approach to Wholeness*, Albany, New York: State University of New York Press.

ROYCE, W.W. (1970) Managing the Development of Large Software Systems: Concepts and Techniques, *Proceedings of IEEE WESCON*, Los Angeles, pp. 1-9.

RYLE, G. (1954) *The Concept of Mind*, London: Hutchinson.

RYLE, G. (1979) Improvisation, in G. Ryle (Ed.), *On Thinking*, London: Blackwell, pp. 121-130.

SANDERS, M. (Ed.) (1998) *The SPIRE Handbook: Better, Faster, Cheaper Software Development in Small Organisations*, Dublin: Centre for Software Engineering Ltd.

SARAPH, J.V., BENSON, P.G., AND SCHROEDER, R.G. (1989) An Instrument for Measuring the Critical Factors of Quality Management, *Decision Sciences*, Vol. 20, No. 4, pp. 810-829.

SAUER, J., WASTELL, D.G., AND HOCKEY, G.R.J. (2000) A Conceptual Framework for Designing Micro-worlds for Complex Work Domains: A Case Study of the Cabin Air Management System, *Computers in Human Behavior*, Vol. 16, No. 1, pp. 45-58.

SAWYER, S. AND GUINAN, P.J. (1998) Software Development: Processes and Performance, *IBM Systems Journal*, Vol. 37, No. 4, pp. 552-569.

SCACCHI, W. (1984) Managing Software Engineering Projects: A Social Analysis, *IEEE Transactions on Software Engineering*, Vol. 10, No. 1, pp. 49-59.

SCHEIN, E.H. (1987) *Process Consultation*, Second Edition, Reading, Massachusetts: Addison-Wesley.

SCHEIN, E.H. (1992) *Organizational Culture and Leadership*, Second Edition, San Francisco: Jossey-Bass.

SCHEIN, E.H. (1996a) Culture: The Missing Concept in Organization Studies, *Administrative Science Quarterly*, Vol. 41, pp. 229-240.

SCHEIN, E.H. (1996b) Kurt Lewin's Change Theory in the Field and in the Classroom: Notes Towards a Model of Managed Learning, *Systems Practice*, Vol. 9, No. 1, pp. 27-47.

SCHEIN, E.H. (1996c) Three Cultures of Management: The Key to Organizational Learning, *Sloan Management Review*, Vol. 38, No. 1, Fall, pp. 9-20.

SCHERTZER, C.B. AND KERNAN, J.B. (1985) More on the Robustness of Response Scales, *Journal of Marketing Research Society*, Vol. 8, No. 4, pp. 261-282.

SCHULTZ, M. AND HATCH, M. (1996) Living with Multiple Paradigms: The Case of Paradigm Interplay in Organizational Culture Studies, *Academy of Management Review*, Vol. 21, No. 2, pp. 529-557.

SCHUMACHER, E.F. (1977) *A Guide for the Perplexed*, New York: Harper and Row.

SCHWABER, K. AND BEEDLE, M. (forthcoming, 2001) *Scrum - Agile Software Development*, Prentice Hall.

SCHWANDT, T.A. (2000) Three Epistemological Stances for Qualitative Inquiry: Interpretivism, Hermeneutics, and Social Constructionism, in N.K. Denzin and Y.S. Lincoln (Eds.), *Handbook of Qualitative Research*, Second Edition, Thousand Oaks, California: Sage, pp. 189-213.

SCHÖN, D.A. (1983) *The Reflective Practitioner: How Professionals Think in Action*, New York: Basic Books.

SCUPIN, R. (1997) The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology, *Human Organization*, Vol. 56, No. 2, pp. 233-237.

SEI (2000a) *SCAMPI, Standard CMMI Assessment Method for Process Improvement: Method Description*, Version 1.0, Technical Report, CMU/SEI-2000-TR-009, Carnegie Mellon University, Software Engineering Institute.

SEI (2000b) *CMMI-SE/SW, Capability Maturity Model – Integrated for Systems Engineering/ Software Engineering*, Version 1.0, Staged Representation, Technical Report, CMU/SEI-2000-TR-018, Software Engineering Institute, Carnegie Mellon University.

SEI (2000c) *CMMI-SE/SW, Capability Maturity Model – Integrated for Systems Engineering/ Software Engineering*, Version 1.0, Continuous Representation, Technical Report, CMU/SEI-2000-TR-019, Software Engineering Institute, Carnegie Mellon University.

SEKARAN, U. (1992) *Research Methods for Business: A Skill Building Approach*, New York: John Wiley & Sons.

SENGE, P.M. (1990) *The Fifth Discipline: The Art and Practice of the Learning Organization*, New York: Doubleday.

SENGE, P.M., KLEINER, A., ROBERTS, C., ROSS, R., ROTH, G., AND SMITH, B. (1999) *The Dance of Change: The Challenges of Sustaining Momentum in Learning Organizations*, New York: Currency/Doubleday.

SHANNON, C.E. AND WEAVER, W. (1949) *The Mathematical Theory of Communication*, Urbana: University of Illinois Press.

SIDDIQI, J. (1994) Challenging Universal Truths of Requirements Engineering, *IEEE Software*, Vol. 11, No. 2, pp. 18-19.

SIMON, H.A. (1991) Bounded Rationality and Organizational Learning, *Organization Science*, Vol. 2, No. 1, pp. 125-134.

SIMON, J.-M., EL EMAM, K., ROUSSEAU, S., JACQUET, E., AND BABEY, F. (1997) The Reliability of ISO/IEC PDTR 15504 Assessments, *Software Process: Improvement and Practice*, Vol. 3, pp. 177-188.

SINGER, J. (1999) Using the American Psychological Association (APA) Style Guidelines to Report Experimental Results, *Proceedings of Workshop on Empirical Studies in Software Maintenance*, Oxford, England, September, NRC 44130, pp. 71-75, National Research Council Canada.

SITKIN, S.B. (1992) Learning Through Failure: The Strategy of Small Losses, *Research in Organizational Behavior*, Vol. 14, pp. 231-266.

SMITH, J.K. (1983) Quantitative versus Qualitative Research: An Attempt to Clarify the Issue, *Educational Researcher*, Vol. 12, pp. 6-13.

SODAN, A.C. (1998) Yin and Yang in Computer Science, *Communications of the ACM*, Vol. 41, No. 4, pp. 103-111.

SOMMERVILLE, I. AND SAWYER, P. (1997) *Requirements Engineering: A Good Practice Guide*, Chichester: John Wiley & Sons.

SOTIROVSKI, D. (2001) Heuristics for Iterative Software Development, *IEEE Software*, Vol. 18, No. 3, pp. 66-73.

SPECTOR, P. (1976) Choosing Response Categories for Summated Rating Scales, *Journal of Applied Psychology*, Vol. 61, No. 3, pp. 374-375.

SPECTOR, P. (1980) Ratings of Equal and Unequal Response Choice Intervals, *The Journal of Social Psychology*, No. 112, pp. 115-119.

SPECTOR, P. (1992) *Summated Rating Scale Construction: An Introduction*, Sage University Paper series on Quantitative Applications in the Social Sciences, series no. 07-082, Newbury Park, California: Sage.

SPSS (1999a) *SPSS Base 9.0: User's Guide*, Chicago, IL: SPSS Inc.

SPSS (1999b) *SPSS Base 9.0: Applications Guide*, Chicago, IL: SPSS Inc.

STAKE, R.E. (1995) *The Art of Case Study Research*, Thousand Oaks, California: Sage.

STAPLETON, J. (1997) *DSDM: Dynamic Systems Development Method*, Harlow, England: Addison Wesley.

STARBUCK, W.H. AND MILLIKEN, F.J. (1988) Executives' Perceptual Filters: What They Notice and How They Make Sense, in D.C. Hambrick (Ed.), *The Executive Effect: Concepts and Methods for Studying Top Managers*, Greenwich, CT: JAI Press, pp. 35-65.

STATA, RAY (1989) Organizational Learning – The Key to Management Innovation, *Sloan Management Review*, Vol. 30, No. 3, pp. 63-74.

STELZER, D. AND MELLIS, W. (1998) Success Factors of Organizational Change in Software Process Improvement, *Software Process – Improvement and Practice*, Vol. 4, No. 4, pp.227-250.

STELZER, D., MELLIS, W., AND HERZWURM, G. (1996) Software Process Improvement via ISO 9000? Results of Two Surveys among European Software Houses, *Proceedings of the 29th Hawaii International Conference on Systems Sciences*, January 3-6, Wailea, Hawaii, USA.

STEVENS J. (1992) *Applied Multivariate Statistics for the Social Sciences*, London: Lawrence Erlbaum.

STEVENS, S.S. (1946) On the Theory of Scales of Measurement, *Science*, Vol. 103, pp. 677-680.

STEVENS, S.S. (1951) Mathematics, Measurement, and Psychophysics, in S.S. Stevens (Ed.) *Handbook of Experimental Psychology*, New York: Wiley.

STEVENS, S.S. (1958) Problems and Methods of Psychophysics, *Psychological Bulletin*, Vol. 55, p. 177-196.

STEVENS, S.S. (1960) Ratio Scales, Partition Scales, and Confusion Scales, in H. Gulliksen and S. Messick (Eds.), *Psychological Scaling: Theory and Applications*. New York: Wiley.

STOKKE, H.E. AND PALMSTRØM, R. (1999) Software Process Improvement through Software Metrics – an ESSI Project, *Telektronikk*, Vol. 95, No. 1, pp. 62-65.

STRAUB, D.W. (1989) Validating Instruments in MIS research, *MIS Quarterly*, Vol. 13, No. 2, pp. 147-169.

STRAUSS, A.L. AND CORBIN, J.M. (1998) *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, Second Edition, Thousand Oaks, California: Sage Publications.

STÅLHANE, T., BORGERSEN, P.C., AND ARNESEN, K. (1997a) In Search of the Customer's Quality View, *Journal of Systems and Software*, Vol. 38, No. 1, pp. 85-93.

STÅLHANE, T., DYBÅ, T., AND PALMSTRØM, R. (1997b) Experience of Introducing Goal-Oriented Measurement, *Proceedings of the 8th International Workshop on Software Technology and Engineering Practice (STEP'97)*, London, July 14-18.

STÅLHANE, T., WEDDE, K.J., AND DYBÅ, T. (1998) Data Driven Improvement for SMEs, *Proceedings of the European Software Process Improvement Conference (EuroSPI'98)*, Gothenburg, Sweden, November 16-18.

SUCHMAN, L.A. (1987) *Plans and Situated Actions: The Problem of Human-Machine Communication*, New York: Cambridge University Press.

SUSMAN, G.I. (1983) Action Research: A Sociotechnical Systems Perspective, in G. Morgan (Ed.), *Beyond Method: Strategies for Social Research*, Newbury Park, California: Sage, pp. 95-113.

SØRUMGÅRD, S. (1997) *Verification of Process Conformance in Empirical Studies of Software Development*, Doctoral Dissertation, Trondheim, Norway: Norwegian University of Science and Technology.

SAARINEN, T. (1996) An Expanded Instrument for Evaluating Information System Success, *Information & Management*, Vol. 31, No. 2, pp. 103-118.

TAGUCHI, G. (1986) *Introduction to Quality Engineering: Designing Quality into Products and Processes*, Tokyo: Asian Productivity Organization.

TAGUCHI, G., ELSAYED, E.A., AND HSIANG, T.C. (1989*) Quality Engineering in Production Systems*, New York: McGraw-Hill.

TASHAKKORI, A. AND TEDDLIE, C. (1998) *Mixed Methodology: Combining Qualitative and Quantitative Approaches*, Applied Social Research Methods Series, Vol. 46, Thousand Oaks, California: Sage.

TAYLOR, F.W. (1911) *The Principles of Scientific Management*, Newton Library Harper & Row.

TENG, J.T.C., JEONG, S.R., AND GROVER, V. (1998) Profiling Successful Reengineering Projects, *Communications of the ACM*, Vol. 41, No. 6, pp. 96-102.

TERZIOVSKI, M. AND SAMSON, D. (2000) The Effect of Company Size on the Relationship between TQM Strategy and Organizational Performance, *The TQM Magazine*, Vol. 12, No. 2, pp. 144-148.

TETENBAUM, T.J. (1998) Shifting Paradigms: From Newton to Chaos, *Organizational Dynamics*, Vol. 26, No. 4, pp. 21-32.

THOMAS, J.B., CLARK, S.M., AND GIOIA, D.A. (1993) Strategic Sensemaking and Organizational Performance: Linkages among Scanning, Interpretation, Action, and Outcomes, *Academy of Management Journal*, Vol. 36, No. 2, pp. 239-270.

THOMAS, M. AND MCGARRY, F. (1994) Top-Down vs. Bottom-Up Process Improvement, *IEEE Software*, Vol. 11, No. 4, pp. 12-13.

THOMAS, R.J. (1994) *What Machines Can't Do*, Berkeley, California: University of California Press.

THORSRUD, E. (1968) Socio-Technical Approach to Job Design and Organizational Development, *Management International Review*, Vol. 8, pp. 120-131.

THORSRUD, E., SØRENSEN, B., AND GUSTAVSEN, B. (1976) Sociotechnical Approach to Industrial Democracy in Norway, in R. Dubin (Ed.), *Handbook of Work Organization and Society*, Chicago: Rand McNally, pp. 648-687.

TICHY, W.F. (1998) Should Computer Scientists Experiment More?, *IEEE Computer*, Vol. 31, No. 5, pp. 32-39.

TICHY, W.F., LUKOWICZ, P., PRECHELT, L., AND HEINZ, E.A. (1995) Experimental Evaluation in Computer Science: A Quantitative Study, *Journal of Systems and Software*, Vol. 28, No. 1, pp. 9-18.

TINGEY, M.O. (1996) Comparing ISO 9000, Malcolm Baldridge, and the SEI CMM for Software: A Reference and Selection Guide, Prentice Hall.

TOULMIN, S. (1996) Concluding Methodological Reflection: Elitism and Democracy among the Sciences, in S. Toulmin and B. Gustavsen (Eds.), *Beyond Theory*, Amsterdam: John Benjamins, pp. 203-225.

TRIST, E. (1981) The Evolution of Socio-Technical Systems: A Conceptual Framework and an Action Research Program, *Occasional papers No. 2*, Toronto, Ontario: Ontario Quality of Working Life Center.

TRIST, E. AND BAMFORTH (1951) Some Social and Psychological Consequences of the Longwall Method of Coal Getting, *Human Relations*, Vol. 4, No. 1, pp. 3-38.

TUKEY, J.W. (1961) Data Analysis and Behavioral Science or Learning to Bear the Quantitative Man's Burden by Shunning Badmandments, in L.V. Jones (Ed.), *The Collected Works of John W. Tukey, Vol III: Philosophy and Principles of Data Analysis: 1949-1964* (1986), Monterey, California: Wadsworth & Brooks/Cole Advanced Books & Software, pp. 391-484.

TUKEY, J.W. (1962) The Future of Data Analysis, in L.V. Jones (Ed.), *The Collected Works of John W. Tukey, Vol III: Philosophy and Principles of Data Analysis: 1949-1964* (1986), Monterey, California: Wadsworth & Brooks/Cole Advanced Books & Software, pp. 187-389.

TUSHMAN, M.L. AND ROMANELLI, E. (1985) Organizational Evolution: A Metamorphosis Model of Convergence and Reorientation, in L.L. Cummings and B.M Staw (Eds.), *Research in Organizational Behavior*, Vol. 7, Greenwich, Connecticut: JAI Press, pp. 171-222.

TYRE, M.J. AND VON HIPPEL, E. (1997) The Situated Nature of Adaptive Learning in Organizations, *Organization Science*, Vol. 8, No. 1, pp. 71-83.

UCHIMARU, K., OKAMOTO, S., AND KURAHARA, B. (1993) *TQM for Technical Groups: Total Quality Principles for Product Development*, Portland, Oregon: Productivity Press.

VAN DE VEN, A.H. (1986) Central Problems in the Management of Innovation, *Management Science*, Vol. 32, No. 5, pp. 590-607.

VAN DE VEN, A.H. AND FERRY, D.L. (1980) *Measuring and Assessing Organization*, New York: John Wiley & Sons.

VAN DE VEN, A.H., ANGLE, H.L., AND POOLE, M.S. (Eds.) (2000) *Research on the Management of Innovation: The Minnesota Studies*, Oxford: Oxford University Press.

VAN DE VEN, A.H., POLLEY, D.E., GARUD, R., AND VENKATARAMAN, S. (1999) *The Innovation Journey*, Oxford: Oxford University Press.

VAN DER BENT, J., PAAUWE, J., AND WILLIAMS, R. (1999) Organizational Learning: An Exploration of Organizational Memory and its Role in Organizational Change Processes, *Journal of Organizational Change Management*, Vol. 12, No. 5, pp. 377-404.

VAN SOLINGEN, R. AND BERGHOUT, E. (1999) *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*, London: McGraw-Hill.

VELLEMAN, P.F. AND WILKINSON, L. (1993) Nominal, Ordinal, Interval, and Ratio Typologies are Misleading, *The American Statistician*, Vol. 47, No. 1, pp. 65-72.

VENKATRAMAN, N. AND RAMANUJAM, V. (1987) Measurement of Business Economic-Performance – An Examination of Method Convergence, *Journal of Management*, Vol. 13, No. 1, pp. 109-122.

VON KROGH, G. AND GRAND, S. (2000) Justification in Knowledge Creation: Dominant Logic in Management Discourses, in G. von Krogh, I. Nonaka, and T. Nishiguchi (Eds.), *Knowledge Creation: A Source of Value*, London: MacMillan, pp. 13-35.

VON KROGH, G., ICHIJO, K., AND NONAKA, I. (2000) *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*, New York: Oxford University Press.

WAGNER, R. (1981) *The Invention of Culture*, Revised and Expanded Edition, Chicago: The University of Chicago Press.

WALSH, J.P. AND UNGSON, G.D. (1991) Organizational Memory, *Academy of Management Review*, Vol. 16, No. 1, pp. 57-91.

WALZ, D.B., ELAM, J.J., AND CURTIS, B. (1993) Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration, *Communications of the ACM*, Vol. 36, No. 10, pp. 63-77.

WASTELL, D.G. (1996) The Fetish of Technique: Methodology as a Social Defence, *Information Systems Journal*, Vol. 6, No. 1, pp. 25-40.

WASTELL, D.G. (1999) The Human Dimension of the Software Process, in J.-C. Derniame, B.A. Kaba, and D.G. Wastell (Eds.), *Software Process: Principles, Methodology, and Technology*, Lecture Notes in Computer Science, Vol. 1500, Berlin: Springer-Verlag, pp. 165-199.

WEBER, M. (2000) *Makt og byråkrati: Essays om politikk og klasse, samfunnsforskning og verdier* (Power and Bureaucracy: Essays about politics, class, social science and values), Third Edition, from "Wirtschaft und Gesellschaft" (1922), "Gesammelte Aufsätze zur Wissenschaftslehre" (1922), and "Gesammelte politische Schriften" (1921), Oslo, Norway: Gyldendal (in Norwegian).

WEICK, K.E. (1979) *The Social Psychology of Organizing*, Second Edition, Reading, Massachusetts: Addison-Wesley.

WEICK, K.E. (1995) *Sensemaking in Organizations*, Thousand Oaks, California: Sage Publications.

WEINBERG, G.M. (1971) *The Psychology of Computer Programming*, New York: Van Nostrand Reinhold.

WEINBERG, G.M. (1998) *The Psychology of Computer Programming*, Silver Anniversary Edition, New York: Dorset House.

WENGER, E. (1998) *Communities of Practice: Learning, Meaning, and Identity*, Cambridge: Cambridge University Press.

WESTGAARD, R.H. AND GUSTAD, S. (1999) A Total Improvement Strategy as a Basis for Software Process Improvement, *Proceedings of the European Software Process Improvement Conference (EuroSPI'99)*, Pori, Finland, 25-27 October.

WICKS, A.C. AND FREEMAN, R.E. (1998) Organization Studies and the New Pragmatism: Positivism, Anti-positivism, and the Search for Ethics, *Organization Science*, Vol. 9, No.2, pp. 123-140.

WIKSTRÖM, S. AND NORMANN, R. (1994) *Knowledge and Value: A New Perspective on Corporate Transformation*, London: Routledge.

WILKINSON, L. (1996) Discussion on the Paper by Hand, in Bartholomew *et al*., Statistics and the Theory of Measurement – Discussion of the Paper by Hand (1996), *Journal of the Royal Statistics Society*, Series A, Vol. 159, Part 3, pp. 473-492.

WINOGRAD, T.A. (Ed.) (1996) *Bringing Design to Software*, with J. Bennett, L. De Young, and B. Hartfield, ACM Press Books, Reading, Massachusetts: Addison-Wesley.

WINOGRAD, T.A. AND FLORES, F. (1986) *Understanding Computers and Cognition: A New Foundation for Design*, Reading, Massachusetts: Addison-Wesley.

WOHLIN, C., RUNESON, P., HÖST, M., OHLSSON, M.C., REGNELL, B., AND WESSLÉN, A. (2000) *Experimentation in Software Engineering: An Introduction*, Boston: Kluwer Academic Publishers.

YIN, R.K. (1994) *Case Study Research: Design and Methods*, Second Edition, Applied Social Research Methods Series, Vol. 5, Thousand Oaks, California: Sage Publications.

YUSOF, S.M. AND ASPINWALL, E. (1999) Critical Success Factors for Total Quality Management Implementation in Small and Medium Enterprises, *Total Quality Management*, Vol. 10, Nos 4&5, pp. 803-809.

ZAHRAN, S. (1998) *Software Process Improvement: Practical Guidelines for Business Success*, Harlow, England: Addison-Wesley.

ZAJAC, G. AND BRUHN, J.G. (1999) The Moral Context of Participation in Planned Organizational Change and Learning, *Administration & Society*, Vol. 30, No. 6, pp. 706-733.

ZELKOWITZ, M.V. AND WALLACE, D.R. (1998) Experimental Models for Validating Technology, *IEEE Computer*, Vol. 31, No. 5, pp. 23-31.

ZENDLER, A. (2001) A Preliminary Software Engineering Theory as Investigated by Published Experiments, *Empirical Software Engineering*, Vol. 6, No. 2, pp. 161-180.

ZIMAN, J.M. (Ed.) (2000) *Technological Innovation as an Evolutionary Process*, Cambridge: Cambridge University Press.

ZUBOFF, S. (1988) *In the Age of the Smart Machine*, New York: Basic Books.

ZULTNER, R.E. (1993) TQM for Technical Teams, *Communications of the ACM*, Vol. 36, No. 10, pp. 79-91.

ZWICK, W.R. AND VELICER, W.F. (1986) Comparison of Five Rules for Determining the Number of Components to Retain, *Psychological Bulletin*, Vol. 99, pp. 432-442.

ØGRIM, L. (1993) Ledelse av systemutviklingsprosjekter: En dialektisk tilnærming (Management of Systems Development Projects: A Dialectic Approach), Doctoral Dissertation, Oslo, Norway: University of Oslo (in Norwegian).